

UNITED STATES PATENT APPLICATION

OF

DIETRICH CHARISIUS,

PETER COAD,

JONATHAN KERN,

AND

MIKHAIL OKRUGIN

FOR

**METHODS AND SYSTEMS FOR OPTIMIZING RESOURCE ALLOCATION BASED
ON DATA MINED FROM PLANS CREATED FROM A WORKFLOW**

Docket No. TS1006

METHODS AND SYSTEMS FOR OPTIMIZING RESOURCE ALLOCATION BASED ON DATA MINED FROM PLANS CREATED FROM A WORKFLOW

Cross-Reference To Related Applications

This application claims the benefit of the filing date of U.S. Provisional Application No. 60/230,054, entitled "Development Tool for Modeling Workflow," filed on September 1, 2000, and U.S. Provisional Application No. 60/296,707, entitled "Improved Development Tool For Modeling Workflow," filed on June 7, 2001, both of which are incorporated herein by reference.

The following identified U.S. patent applications are also relied upon and are incorporated by reference in this application:

U.S. Patent Application No. _____, entitled "Methods and Systems for Integrating Process Modeling and Project Planning," bearing attorney docket no. TS1000, and filed on the same date herewith;

U.S. Patent Application No. _____, entitled "Methods and Systems for Improving a Workflow Based on Data Mined from Plans Created from the Workflow," bearing attorney docket no. TS1001, and filed on the same date herewith; and

U.S. Patent Application No. _____, entitled "Methods and Systems for Animating a Workflow and a Project Plan," bearing attorney docket no. TS1005, and filed on the same date herewith.

Field Of The Invention

The present invention relates to a method and system for developing a workflow and creating project plans from the workflow. More particularly, the invention relates to methods and systems for optimizing resource allocation and resource profiles used in resource allocation based on data mined from plans created from a workflow.

Background Of The Invention

To become more efficient and competitive, businesses and industries have striven to capture and streamline the business processes or workflows they use to operate and manage their respective enterprises. In general, a workflow is a model of a process. More specifically, a workflow can be viewed as a structured set of activities designed to produce a specific output for a particular customer (internal or external to an enterprise) or market. Although conventional

software tools define the steps performed by the workflow, conventional tools do not schedule the resources (e.g., the people, equipment, or software technologies) responsible for completing each activity. Conventional tools also do not prepare a timeline identifying the beginning or end of each activity. Thus, conventional tools do not prepare a schedule for completing the workflow.

Businesses and industries also use other conventional software tools, such as Microsoft Project™, to build and manage a project plan for their respective enterprises. A plan represents an instance of the workflow. More specifically, a plan can be viewed as a working schedule for a project to produce a product or artifact, such as a computer, bicycle, or software build, for the respective enterprise. These other conventional software tools typically display the working schedule in the form of an interactive Gantt chart, i.e., a chart to which the user can make updates. A Gantt chart is the linear, time-based representation of a project schedule, usually laid out on a horizontal plane where the times/dates increase to the right. These Gantt charts show the temporal relationships between the different tasks in a project, where the tasks are arranged along the vertical axis. Gantt charts are typically used to lay out an initial plan/timeline for the project, and then to track the actual progress of a project from start to finish. The modern software-based Gantt chart also identifies the resource(s) responsible for completing each task of the plan, the dependencies between the tasks, and, once the project has begun, the status of each task.

The conventional tools that support the building and managing of a project plan, however, do not provide direct links between projects and the workflows or business processes that the enterprise has defined and seeks to implement to gain business advantage and economies of efficiencies. Likewise, the conventional tools that enterprises use to define and manage workflows are not linked to project plans. Because both workflows and project plans do not exist on the same tool, workflows and project plans cannot be integrated or synchronized to keep the workflows and project plans “in step” with each other. Thus, there is a need in the art for a tool that avoids the limitations of these conventional software tools.

Summary Of The Invention

Methods and systems consistent with the present invention provide a workflow modeling and project planning integration tool that overcomes the limitations of conventional tools. The integration tool of the present invention has a Client Interface that saves an enterprise or

organization significant development costs and increases its operating efficiency by improving profiles of resources associated with a workflow based on data mined from plans created from the workflow. By improving the profiles of the resources, the integration tool is able to optimize the automatic assignment or allocation of the resources to tasks of a plan when the plan is created from the workflow. The data mined by the Client Interface includes a task found in each plan that has an auto-assigned resource and has a replacement resource. The auto-assigned resource is a resource that the Client Interface automatically assigned to handle a role of the task when the Client Interface created the task from an activity of the workflow. The replacement resource is another resource that was manually assigned by an enterprise affiliate using the Client Interface to replace the auto-assigned resource. The Client Interface improves a profile associated with the auto-assigned resource by removing a skill or decreasing a strength of the skill in the profile, where the skill is associated with the role of the mined task. The Client Interface improves a profile associated with the replacement resource by adding a skill or increasing a strength of the skill in the profile, where the skill is also associated with the role of the mined task. Thus, by improving the profiles for the auto-assigned resource and the replacement resource, the Client Interface is able to more effectively allocate resources to plans that are subsequently created from the workflow by the enterprise affiliate, saving the enterprise affiliate time and effort needed to manually override any resources that were not optimally automatically assigned.

In accordance with methods consistent with the present invention, a method is provided in a data processing system. The data processing system has a workflow that models a process. The method generates a plan to perform an instance of the process, the plan having tasks performed by resources. Each resource has capabilities that are considered when generating the plan to ensure that, for each task, a suitable one of the resources is selected to perform the task. The method also receives modification information indicating that the capabilities of one of the resources has changed, and assigns the resources to the tasks to generate a new plan by using the received modification information.

In accordance with methods consistent with the present invention, a method is provided in a data processing system. The data processing system has a workflow that models a process and has a plurality of plans generated from the workflow that reflect instances of the process. The method receives a request to generate a new plan, examines the plurality of plans to determine a number of the plurality of the plans that have been modified per a modification

since the modified plans were created, determines whether the number of plans exceeds a predetermined threshold, and when it is determined that the number of plans exceeds the predetermined threshold, generates the new plan such that the new plan incorporates the modification.

5 In accordance with articles of manufacture consistent with the present invention, a computer-readable medium is provided. The computer-readable medium contains instructions for controlling a data processing system to perform a method. The data processing system has a workflow that models a process. The method comprises the step of generating a plan to perform an instance of the process. The generated plan has tasks performed by resources. Each resource
10 has capabilities that are considered when generating the plan to ensure that, for each task, a suitable one of the resources is selected to perform the task. The method also comprises the steps of receiving modification information indicating that the capabilities of one of the resources has changed, and assigning the resources to the tasks to generate a new plan by using the received modification information.

15 In accordance with articles of manufacture consistent with the present invention, a computer-readable medium is provided. The computer-readable medium contains instructions for controlling a data processing system to perform a method. The data processing system has a workflow that models a process and has multiple plans generated from the workflow that reflect instances of the process. The method comprises the steps of receiving a request to generate a new plan, examining the plurality of plans to determine a number of the plurality of the plans
20 that have been modified per a modification since the modified plans were created, determining whether the number of plans exceeds a predetermined threshold, and when it is determined that the number of plans exceeds the predetermined threshold, generating the new plan such that the new plan incorporates the modification.

25 Other systems, methods, features and advantages of the invention will be or will become apparent to one with skill in the art upon examination of the following figures and detailed description. It is intended that all such additional systems, methods, features and advantages be included within this description, be within the scope of the invention, and be protected by the accompanying claims.

Brief Description Of The Drawings

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an implementation of the present invention and, together with the description, serve to explain the advantages and principles of the invention. In the drawings:

Fig. 1 depicts a data processing system suitable for practicing methods and systems consistent with the present invention;

Fig. 2 depicts an architectural overview of the workflow modeling and project planning integration tool used to perform methods and systems consistent with the present invention;

Fig. 3 depicts a flow diagram illustrating the high-level process performed by the tool of Fig. 2 in accordance with methods and systems consistent with the present invention;

Fig. 4 depicts an exemplary document workflow modeled by an enterprise affiliate using the tool of Fig. 2;

Fig. 5 depicts an exemplary task workflow modeled by an enterprise affiliate using the tool of Fig 2;

Fig. 6 depicts another exemplary workflow modeled by an enterprise affiliate using the tool of Fig 2;

Fig. 7 depicts a timeline of the task created from the workflow of Fig. 4;

Fig. 8 depicts a timeline of the task created from the workflow of Fig. 5;

Fig. 9 depicts a timeline of the task created from the workflow of Fig. 6;

Figs. 10-12 depict the execution of the plan depicted in Fig. 7;

Figs. 13-16 depict the execution of the plan depicted in Fig. 8;

Figs. 17-21 depict the execution of the plan depicted in Fig. 9 following the default path;

Figs. 22-27 depict the execution of the plan depicted in Fig. 9 following the non-default path;

Figs. 28A-C depict a flow diagram illustrating the creation or retrieval of a workflow by the tool of Fig. 2;

Fig. 29 depicts an exemplary user interface of the tool of Fig. 2 used to begin creating or retrieving a workflow;

Fig. 30 depicts an exemplary user interface of the tool of Fig. 2 used to enter the name of a new workflow group;

Fig. 31 depicts an exemplary user interface of the tool of Fig. 2 used to begin creating a new workflow;

Fig. 32 depicts an exemplary user interface of the tool of Fig. 2 used to enter the name of a new workflow;

Fig. 33A-C depict an exemplary workflow definition file produced by the tool of Fig. 2 for the workflow depicted in Fig. 6;

Fig. 34 depicts an exemplary user interface of the tool of Fig. 2 used to manage a workflow;

Fig. 35 depicts an exemplary user interface of the tool of Fig. 2 used to add a new role to a workflow;

Fig. 36 depicts an exemplary user interface of the tool of Fig. 2 used to select an artifact type;

Fig. 37 depicts an exemplary user interface of the tool of Fig. 2 used to enter condition properties for a document-oriented artifact;

Fig. 38 depicts an exemplary user interface of the tool of Fig. 2 used to enter condition properties for a script-oriented artifact;

Fig. 39 depicts an exemplary user interface of a script editor for the tool of Fig. 2;

Fig. 40 depicts an exemplary user interface of the tool of Fig. 2 used to modify the properties of a workflow activity;

Figs. 41A and B depict a flow diagram illustrating the creation of a plan from a workflow;

Fig. 42 depicts an exemplary user interface of the tool of Fig. 2 used to create a new plan group;

Fig. 43 depicts an exemplary user interface of the tool of Fig. 2 displaying the available plan groups;

Fig. 44 depicts an exemplary user interface of the tool of Fig. 2 used to enter a plan name;

Fig. 45 depicts an exemplary user interface of the tool of Fig. 2 used to enter the working schedule;

Fig. 46 depicts an exemplary user interface of the tool of Fig. 2 used to enter the scheduled start and end times for the plan;

Fig. 47 depicts an exemplary workflow definition file produced by the tool of Fig. 2 for the workflow of Fig. 5 is created;

Fig. 48 depicts an exemplary plan definition file created from the workflow definition file of Fig. 47;

Fig. 49 depicts an exemplary user interface of the tool of Fig. 2 used to assign users to a plan;

Fig. 50 depicts an exemplary user interface of the tool of Fig. 2 used to edit the properties of a plan;

Fig. 51 depicts a timeline of the task created from the workflow of Fig. 5;

Fig. 52 depicts an exemplary timeline of the tool of Fig. 2 used to activate a plan;

Fig. 53 depicts a flow diagram illustrating the addition of a resource by the tool of Fig. 2;

Fig. 54 depicts an exemplary user interface of the tool of Fig. 2 used to add a resource;

Fig. 55 depicts an exemplary user interface of the tool of Fig. 2 used to receive LDAP access information;

Fig. 56 depicts an exemplary resource file created by the tool of Fig. 2;

Fig. 57 depicts a flow diagram illustrating the management of an activated plan;

Fig. 58 depicts a timeline of the task created from the workflow of Fig. 5;

Fig. 59 depicts an exemplary plan definition file created from the workflow of Fig. 5;

Figs. 60, 62, 64 and 66 depict the actual timeline showing the execution of the plan depicted in Fig. 58;

Figs. 61, 63, and 65 depict the properties of the executing tasks of Figs. 62, 64, and 66;

Figs. 67A through 67E depict a flowchart illustrating an exemplary process performed by the Client Interface to improve a profile of any resource that has been previously assigned to a role of a task in a plan created from a workflow;

Fig. 68 depicts an excerpt of an exemplary workflow definition file created and stored by the Client Interface;

Fig. 69 depicts an exemplary graphical representation of the workflow definition file in Fig. 68 displayed by the Client Interface;

Fig. 70 depicts an exemplary workflow roles file created by the Client Interface to store profiles of roles assigned to activities in the workflow definition file in Fig. 68;

Fig. 71 depicts an exemplary resource file created by the Client Interface to store resource profiles that have been defined by an enterprise affiliate using the Client Interface;

Fig. 72 depicts an exemplary skills file created by the Client Interface to store skills that may be assigned to a role or a resource by the Client Interface;

Figs. 73A-C depict an excerpt of an exemplary plan definition file created by the Client Interface in response to an enterprise affiliate manually assigning a resource to a role associated with a "Get Parts" task in the plan definition file; and

Fig. 74 depicts an excerpt of another exemplary plan definition file that is an earlier version of the plan definition file in Figs. 73A-C and that was created by the Client Interface from the workflow definition file in Fig. 68.

Detailed Description Of The Invention

Methods and systems consistent with the present invention provide an integrated workflow modeling and project planning integration tool that improves the efficiency and reduces the operating cost of an enterprise or business conglomerate. Contrary to conventional tools that do not allow a user to integrate a workflow and a project plan, the integration tool allows a user to model a business process or workflow, to create and activate a project plan based on the workflow, and to track the progress of the activated project plan. The tool also allows the workflow to be reused to create more than one project plan based on the workflow. The tool simultaneously manages the execution of the plans. Moreover, the integration tool may include a Web-based "Distributed Authoring and Versioning" (WebDAV) server that operates as a virtual file system for computers on a network. With the WebDAV server, more than one user on different computer systems may view the same workflow or project plan, monitor the progress of an activated project plan, or simultaneously create and activate different plans from the same workflow.

System Overview

While methods and systems consistent with the present invention may apply to any enterprise, they will be further described below with reference to the software industry to provide clarity. More particularly, methods and systems consistent with the present invention will be described with reference to a software development business process that is applicable to the software industry.

Fig. 1 depicts a data processing system 100 suitable for practicing methods and systems consistent with the present invention. Data processing system 100 includes a group of computers 102a, 104, and 106 that are connected via a network 108. Network 108 may be any known physical or wireless network capable of supporting a data transmission between two

computer systems, such as a Local Area Network (LAN), a Wide Area Network (WAN), the Internet, or leased phone lines.

As further explained herein, computer 102a may actually be one of multiple computers (i.e., computers 102a and 102n) used by affiliates of an enterprise or business conglomerate to communicate with one another via network 108. The enterprise affiliates may be employees, managers, administrators, suppliers, customers, or other users within the enterprise who may need to create, view, or receive information regarding an activated project plan in accordance with methods and systems consistent with the present invention.

Each computer 102a, 104, and 106 includes a memory (110, 112, and 114, respectively), a secondary storage device (116, 118, and 120, respectively), an I/O device (122, 124, and 126, respectively), and a processor (128, 130, and 132, respectively). Memory 110 in computer 102a includes a Client Interface 134 to a Web-based "Distributed Authoring and Versioning" (WebDAV) server 140 in memory 112. Client Interface 134 has Process and Plan modules 136 that collectively allow an enterprise affiliate to create a reusable workflow and to create and activate a project plan based on the reusable workflow.

Memory 110 in computer 102a also includes a target processor interpreter, such as a Java™ Virtual Machine 138. To permit the Client Interface 134 to run on most any computer, the Client Interface 134 may be developed using the Java™ Programming Language. Thus, Client Interface 134 may include Java™ bytecodes. The Java™ Virtual Machine 138 interprets the Java™ bytecodes of the Client Interface 134 so that the Client Interface 134 may execute on computer 102a.

The WebDAV server 140 in memory 112 of computer 104 operates as a virtual file system for computers 102a, 102n, and 106 on the network 108. To operate as a virtual file system, WebDAV Server 140 communicates on the network 108 using the WebDAV protocol, and stores files on WebDAV storage 142. In one implementation, WebDAV storage 142 may be a known database, such as Oracle, DB2, MS Structured Query Language (SQL) storage, or any Java Database Connectivity (JDBC)-compliant database. In this implementation, WebDAV Server 140 includes a database management system (DBMS) or a JDBC interface to control access to the WebDAV storage 142.

In accordance with methods and systems consistent with the present invention, two separate enterprise affiliates using their respective Client Interfaces 134 on their respective computers 102a and 102n may independently request and view the same workflow or plan

stored on WebDAV Storage 142. In addition, the Client Interface 134 allows any enterprise affiliate to create, delete, move, and copy workflows, project plans, and associated roles/resource lists on WebDAV server 140. Furthermore, properties of a process, a plan, or a task of a plan may be added, located, or changed on WebDAV Storage 142 by Client Interface 134 using known methods of the WebDAV protocol.

The WebDAV protocol is a set of known extensions to the standard HyperText Transfer protocol (HTTP) that allows enterprise affiliates using client computers 102a and 102n to collaboratively store, edit, and manage files remotely on a Web Server, such as WebDAV Server 140 using network 108. As known to one skilled in the art, HTTP defines how messages sent to or from a Web server on the Internet are formatted and transmitted. HTTP also defines what actions a Web server or Web browser of a computer on the Internet should take in response to various commands submitted as part of an HTTP message.

The WebDAV protocol defines a WebDAV resource to be a collection (e.g., a directory or folder on WebDAV Storage 142) or a collection member (e.g., a file or Web page on WebDAV Storage 142). Each WebDAV resource has a content file and properties associated with the content file. The properties include the creation date, the author, and the access rights for the WebDAV resource. The WebDAV protocol specifies the methods to create, delete, move, and copy a WebDAV resource. It also specifies the methods to add, find, or change a property of a WebDAV resource. The WebDAV protocol and the HTTP extensions that comprise the WebDAV protocol are more clearly described in the following reference, which is incorporated herein by reference: HTTP Extensions For Distributed Authoring -- WebDAV, RFC 2518, Standards Track, Proposed Standard, February 1999, available at <http://andrew2.andrew.cmu.edu/rfc/rfc2518.html>.

Memory 114 in computer 106 includes a Tool Server 144. The Tool Server 144 includes a WebDAV proxy 146 and Management Modules 148. WebDAV proxy 146 mediates communication between the Client Interface 134 and the WebDAV server 140. The messages or requests directed to the WebDAV server 140 from the Client Interface 134 are initially sent to the WebDAV proxy 146, as discussed in detail below. The WebDAV proxy 146 passes these messages and requests to the Management Modules 148. Each of the Management Modules 148 is configured to inform the WebDAV proxy 146 when a message or request has been serviced. If none of the Management Modules 148 services the message or request, then the WebDAV proxy 146 sends the message or request to the WebDAV Server 140 via the network 108. If, on

the other hand, the Management Modules 148 are able to service the message or request, the message or request is not sent to the WebDAV Server 140. The types of messages or requests serviced by the Management Modules 148 include activating a project plan, notifying various enterprise affiliates assigned to each task of the plan, and managing the execution of the plan to the enterprise affiliates.

In another implementation, memory 114 in computer 106 includes a WebDAV servlet (not shown), which is an application designed to perform as a WebDAV Engine in lieu of WebDAV Server 140. The WebDAV servlet may be started by and executed within the Tool Server 144. In this implementation, WebDAV servlet is persistent. Thus, once WebDAV servlet is started, it stays in memory and can fulfill multiple requests. WebDAV servlet is configured to control access to WebDAV Storage 142, which may be included in secondary storage 120 in computer 106.

Memory 114 in computer 106 also includes a target processor interpreter, such as a Java™ Virtual Machine 150. As with the Client Interface 134 on computer 102a, the Tool Server 144 includes Java™ bytecodes that the Java Virtual Machine 150 interprets so that the Tool Server 144 may execute on computer 106.

In another implementation, the data processing system 100 may operate in a local host configuration rather than across the network 108. In this implementation, the memory 110 of computer 102a may include the Tool Server 144 and the WebDAV Server 140. In addition, the secondary storage device 116 may include the WebDAV Storage 142.

Although aspects of the present invention are described as being stored in memory, one skilled in the art will appreciate that these aspects can also be stored on or read from other types of computer-readable media, such as secondary storage devices, like hard disks, floppy disks, or CD-ROM; a carrier wave from a network, such as Internet; or other forms of RAM or ROM.

Fig. 2 depicts a functional architectural overview of the workflow modeling and project planning integration tool 200 used to integrate workflow modeling and project planning. As shown in Fig. 2, the tool 200 includes the Client Interface 134 as well as the Tool Server 144. Although part of the same tool 200, the Client Interface 134 and the Tool Server 144 may be located on different computer systems, as discussed above.

The Client Interface 134 includes a Virtual File System (“VFS”) Interface 202 that is configured to allow the Client Interface 134 to connect to the secondary storage device 116 for local file access or to connect to the WebDAV Storage 142 via the WebDAV proxy 146 for

virtual file access. To allow the WebDAV proxy 146 to mediate communication between the Client Interface 134 and the WebDAV Storage 142, the VFS Interface 202 is configured to send the virtual file access requests from the Client Interface 134 to a Uniform Resource Locator (URL) or network address for the WebDAV proxy 146. For example, the URL for the WebDAV proxy 146 may be "http://www.ToolServer.com/WebDAVproxy." A URL typically consists of an access protocol (e.g., *http*), a domain name (e.g., *www.ToolServer.com*), and, optionally, the path to a file or resource residing on that server (e.g., *WebDAVproxy*). If the Tool Server 144, where the WebDAV proxy 146 is located, has an IP address of 192.168.5.1 and an assigned port address of 8088, then the URL for the WebDAV proxy translates to "http://192.168.5.1:8088/WebDAVproxy."

As discussed above, the VFS Interface 202 initially sends the requests that the Client Interface 134 directs to the WebDAV Storage 142 to the WebDAV proxy 146. The WebDAV proxy 146 sends these requests to the Management Modules 148. After the Management Modules 148 review these requests, the WebDAV proxy 146 sends the request to the WebDAV server 140 if the Management Modules 148 do not respond to the requests from the Client Interface 134. If the request is to be sent to the WebDAV server 140, the Tool Server 144 directs the request to a URL or network address for the WebDAV server 140.

The Client Interface 134 also includes a module loader 204 to load the Process and Plan modules 136. As one skilled in the art will appreciate, Client Interface 134 may be developed so that the functionality provided by Process and Plan modules 136 is not loaded by a known module loader 204, but integrally incorporated within the element corresponding to the Client Interface 134. The Process and Plan modules 136 produce the requests to store or modify the various client files on the WebDAV storage 142. As further described below, the various types of client files include a condition model, a user profile, a resource profile, a workflow definition file, and a plan definition file. Each of these files has properties defined in accordance with the WebDAV protocol. The various types of client files follow a schema or document type definition that is known to the Tool Server 144 so that the Tool Server 144 can identify the type of client file sent by the Client Interface 134 and intercepted by the WebDAV Proxy 146. In addition, each type of client file has a unique identifier, such as a URL network address, which the Tool Server 144 may use to locate the associated client file for processing. The various types of client files are discussed in context with the general description of the Process and Plan modules 136 and also further discussed with the implementation details of creating a workflow

and a project plan from the workflow. Although XML files are used to represent the client files used with methods and systems consistent with the present invention, one skilled in the art will recognize that any file type can be used to represent the client files.

The Process and Plan Modules 136 include a Resource Manager Module 206, an Activity I/O Condition Designer Module 208, a Process Designer Module 210, a Project Plan Manager Module 212, and a Task Tracker Module 214. The Resource Manager Module 206 allows an enterprise affiliate with system administrative privileges or permissions, such as a project manager, to create, modify, and store a user profile for an enterprise affiliate. The user profile identifies the access control rights that are associated with the enterprise affiliate, such as whether the enterprise affiliate may create a project plan based on a workflow or whether the enterprise affiliate is limited to viewing an existing workflow or plan. When the Client Interface 134 sends a request to the WebDAV Server 140 to store the user profile, the Client Interface 134 may specify that the user profile be stored with a unique identifier so that the Tool Server 144 may later locate the user profile for further processing. For example, the Client Interface 134 may request that the unique identifier be a location or URL where the user profile is to be stored on the WebDAV Storage 142. If the unique identifier is stored as a property of the user profile on the WebDAV storage 142, the Client Interface 134 sends a request to the WebDAV Server 140 to set the value of the property.

The Resource Manager Module 206 also allows an enterprise affiliate to create, modify, and store the role profiles that may be assigned to an activity of a workflow that is modeled using the tool 200. The role profile identifies a group of resources that may be assigned to complete a task created from the activity. The role profile is a type of client file that the Client Interface 134 may store on WebDAV storage 142 with a unique identifier (e.g., a URL for the role profile) to locate the role profile at a later time. A role profile may include a Rolename that represents a “capability” or “skill set” for the role. For example, using methods and systems consistent with the present invention, an enterprise affiliate may identify one of the following Rolenames to the Resource Manager Module 206 so that the associated role profiles are later available to assign when defining a software development process: Manager, Analyst, Software Architect, Software Developer, Tester, Hardware Architect, and Editor.

In addition to the above, the Resource Manager Module 206 further allows an enterprise affiliate to create, modify, and store the resource profiles (e.g., the person, equipment, or systems, such as a development facility) that may be assigned to a task of a plan created from a

workflow. The resource profile includes a resource ID and a unique identifier for the role profile so that the Client Interface 134 may communicate to the Tool Server 144 that the identified resource has skills or capabilities corresponding to the role profile. For example, when the resource is a person, the Tool Server 144 may recognize that the person can play a given role (e.g., Analyst) in a specific activity in a workflow (e.g., Software Development Process) based on the skills or capabilities required by the role assigned to the activity to be performed.

The Activity I/O Condition Designer Module 208 allows an enterprise affiliate, such as a manager, to define a condition model, i.e., an input condition or an output condition, for an activity of a workflow. The Activity I/O Condition Designer Module 208 stores the condition model with a unique identifier so that the Tool Server 144 may later locate the condition model for processing, such as when a task of a plan is created from the activity of the workflow, as explained below.

As discussed above, there are two types of workflows: a document workflow and a task workflow. Similarly, there are two types of conditions: a document-type condition and a logic-type condition. The Activity I/O Condition Designer Module 208 allows the enterprise affiliate to create a condition model based on one of these two condition types. The Activity I/O Condition Designer Module 208 also allows the enterprise affiliate to assign a document-type condition model or a logic-type condition model to an activity when creating the activity in a workflow. Each document-type and logic-type condition model has properties defined by the enterprise affiliate that created the respective condition model using the Activity I/O Condition Designer Module 208.

The properties of the document-type condition model include a location property (e.g., a URL) identifying the location of the document or artifact being monitored. Thus, when executing a task based on an activity, the Client Interface 134 uses the location property to notify the resource responsible for the task where to find the document or artifact so that the resource may complete its task.

Another property of the document-type condition model is a state property that indicates the allowable states of the document or artifact. For example, the document may have the states "DRAFT" and "APPROVED." When creating the workflow, the enterprise affiliate assigns one of these allowable states as a condition for entry into or exit from the activity (or the task created from the activity). When the task is activated, the Workflow Engine 222 evaluates whether the

state property of the document condition satisfies the input or output condition of the activated task before starting or closing the task.

When creating a logic-type condition model, Activity I/O Condition Designer Module 208 allows the enterprise affiliate to define the properties shown in Table 1.

5

TABLE 1

Property	Description
Name	The name used to identify the condition.
Description	A description of the condition.
When to Check	This section identifies when and/or how often the condition should be checked.
• Abs. Time	The condition is checked when this absolute time (calendar time) arrives.
• Period	Integer expression in Javascript that defines the periodicity of condition check, where a “1” means once a minute. (If absolute time is also specified, then the condition should be checked when the absolute time arrives and periodically thereafter.)
• URL Change	The condition is checked after URL undergoes a property or content change.
• Task Change	The condition is checked when the task that is specified during plan creation changes its state (<i>e.g.</i> , starts, finishes).
• Any Request	The condition is checked when any HTTP request is detected.
Script	The script to run when the condition is met. The script must return “true” or “false” (a Boolean).

When a plan is created from a workflow, the Client Interface 134 uses the logic-type condition model to generate a logic-type condition for entry/exit and the script (*e.g.*, logic element) to be performed to determine if the condition is met. For example, the enterprise

affiliate may indicate to the Activity I/O Condition Designer Module 208 that the condition is to check if the task is complete and that the logic to be performed is to check the status property of the task. In this case, the user or resource assigned to this task must notify the Client Interface 134 that the task is complete. In another example, the enterprise affiliate may indicate to the

5 Activity I/O Condition Designer Module 208 that the condition is to check if the task is complete and that the logic to be performed is to check for the existence of a file in a specific directory folder on WebDAV Storage 142 in order to determine if the task is complete. In this case, the user or resource assigned to this task must create or move a file into the specific directory folder to indicate that the task is complete.

10 The Process Designer Module 210 allows an enterprise affiliate to create, modify, and store a workflow. When the enterprise affiliate indicates to Process Designer Module 210 that the modeled process is to be saved, Process Designer Module 210 produces a workflow definition file based on the modeled workflow object. Client Interface 134 then sends as the workflow definition file as a client file to WebDAV Server 140 to be stored on WebDAV

15 Storage 142. Each workflow definition file produced by Process Designer Module 210 includes a unique identifier (e.g., a URL for the workflow definition file) so that Tool Server 144 may later locate the workflow definition file corresponding to the modeled workflow for further processing.

Project Plan Manager Module 212 allows an enterprise affiliate to create and activate a

20 project plan from a workflow definition file. In general, upon request to create a project plan, Project Plan Manager Module 212 sends a query message to the WebDAV Server 140 for the workflow definition files contained in WebDAV Storage 142. As further described below, Project Plan Manager Module 212 receives the workflow definition files, allows the enterprise affiliate to select one of the workflow definition files to create a project plan, and then produces

25 a plan definition file based on the selected workflow definition file. When instructed to save the plan by the enterprise affiliate, Project Plan Manager Module 212 sends the plan definition file as a client file to WebDAV Server 140 to be stored on WebDAV Storage 142. Each plan definition file produced by Process Designer Module 210 includes a unique identifier (e.g., a URL for the plan definition file) so that Tool Server 144 may later locate the workflow

30 definition file corresponding to the modeled workflow for further processing.

The Task Tracker Module 214 allows an enterprise affiliate to view the tasks of an activated project plan that are assigned to a specific role, to activate a task of the project plan

(e.g., indicate actual start time to Client Interface 134), to open or check-out a document artifact needed to accomplish the task, to close or check-in the document artifact after accomplishing the task, and to indicate that the task is completed.

The Tool Server 144 includes a module loader 216 to load the Management Modules 148. Similar to the Client Interface 134, the Tool Server 144 may be developed so that the functionality provided by Management Modules 148 is not loaded by a known module loader 216, but integrally incorporated within the element corresponding to the Tool Server 144. Management Modules 148 include a User Authorization Module 218, a Resource/Role Management Module 220, and a Workflow Engine 222. The Workflow Engine 222 includes a Project Plan Management Module 224 and a Project Task Management Module 226.

When the Client Interface 134 requests access to a client file on the WebDAV Storage 142, the User Authorization Module 218 verifies that the enterprise affiliate making the request has a user profile on the WebDAV Storage 142 with the proper authorization or permission to access the requested client file. The User Authorization Module 218 may be connected to a Light Directory Access Protocol (LDAP) Import Module 228, which follows a known LDAP protocol to allow the User Authorization Module 218 to obtain existing user profiles from another computer on network 108. As known to those skilled in the art, an LDAP protocol is based on "entries," where an entry is a collection of attributes that have a "distinguished name" (DN). According to the LDAP protocol, directory entries are arranged in a hierarchical tree-like structure that reflects political, geographic, and/or organizational boundaries. For example, entries representing countries may appear at the top of the tree. The entries below the countries may represent states or national organizations. Below the states or national organizations may be entries representing people (e.g., user profiles), organizational units, printers, documents, or any other accessible entity. Each level in the hierarchical tree-like structure for the directory entries may be identified by a known standardized keyword, such as "CN" for the common name of the entry (e.g., user profile), "L" for locality name, "ST" for state or province name, "O" for organization name, "OU" for organizational unit name, and "C" for country name. The LDAP Import Module 228 uses a DN to refer to the entry unambiguously via a concatenation of the hierarchical tree-like structure. After user profiles are retrieved by the User Authorization Module 218 via the LDAP import module 228, the user profiles may then be stored on the WebDAV Storage 142 by a request from the Client Interface 134.

The Resource/Role Management Module 220 reviews requests from an enterprise affiliate to assign a resource to a plan (e.g., to assign a user to a task of the plan). The Resource/Role Management Module 220 checks the resource profile corresponding to the assigned resource on the WebDAV Storage 142 to verify that the resource is not overloaded.

For example, the Resource/Role Management Module 220 determines whether a resource is already assigned to another task in another plan during the same time frame, thus preventing it from being able to complete one of the tasks to which it is assigned. The Resource/Role Management Module 220 may also be connected to the LDAP Import Module 228 to allow the Resource/Role Management Module 220 to obtain existing resource profiles from another computer on network 108. The resource profiles may also be stored on WebDAV Storage 142 by a request from Client Interface 134.

The Workflow Engine 222 reviews requests to activate, deactivate, or update a plan. For example, a request to update a plan occurs if the enterprise affiliate who is an owner of a task indicates in its request that the task is complete. The Workflow Engine 222 also manages the execution of the activated plans.

High-Level Process

Fig. 3 depicts a flow diagram illustrating the high-level process performed by the workflow modeling and project planning integration tool in accordance with methods and systems consistent with the present invention. Initially, the tool creates or retrieves a workflow (step 302). The tool then displays the workflow (step 304). The workflow comprises a set of activities that represents the steps to be performed as part of a plan executed from the workflow. Each activity has an activity description and at least one role responsible for the activity. The activity description indicates what step is to be performed by the role.

There are two types of workflows: a document workflow and a task workflow. In a document workflow, the state of one document (or, more generally, any item or artifact) is monitored by the activities of the workflow. Thus, a document workflow cannot usually have parallel activities, which would require the parallel activities to monitor the states of more than one artifact or would require the parallel activities to monitor different states of the same artifact simultaneously. The document is in one state at a time. Fig. 4 depicts an exemplary document workflow 400. As shown, the workflow 400 includes a start element 402, an end element 404, and two activities, "Step 1" 406 and "Step 2" 408. Because "Step 1" 406 occurs directly before

“Step 2” 408, “Step 1” 406 is the “predecessor activity” to “Step 2” 408. Similarly, “Step 2” 408 is the “successor activity” to “Step 1” 406. The workflow 400 is used to monitor the state of Artifact 410. In particular, in “Step 1” 406, the state of Artifact 410 is “State 1” 412, in “Step 2” 408, the state of Artifact 410 is “State 2” 414, and at the end 404 of the workflow, the state of Artifact 410 is “Complete” 416.

A task workflow, on the other hand, typically has no limitations regarding the number of artifacts that may be monitored or modified by each activity of the workflow to achieve or contribute to the business process goal, such as an auditing process that determines if multiple accounts are balanced properly. Fig. 5 depicts an exemplary task workflow 500. The task workflow 500 includes a start element 502, an end element 504, two serial activities 506 and 508 and two parallel activities 510 and 512. The workflow also includes two synch bars 514 and 516, which are used to connect the ends of parallel activities. Contrary to the document workflow, the task workflow allows for parallel activities.

Another exemplary workflow 600 is depicted in Fig. 6. The workflow 600 includes a start element 602 and an end element 604. The first activity of the workflow 600 is “Get Parts” 606, which is followed by a logic activity, “L or Rt Handed?” 608. Logic activities have two successor activities: a “default activity” and a “non-default activity.” As the name implies, the workflow generally follows the path of the default activity unless a condition is met in the logic activity, as discussed in detail below. In Fig. 6, the default activity is “Right” 610. The non-default activity is “Left” 612, which is followed by another activity “Left Special” 614. The default path is represented as a solid connector 616 while the non-default path is represented as a dotted connector 618. One skilled in the art, however, will recognize that any visible difference in the connectors, e.g., a change in type, color, shading, etc., may be used to represent both the default path as well as the non-default path. Both the default activity 610 and the non-default activities 612 and 614 are followed by another activity, “Complete Assembly” 620.

Returning to Fig. 3, the next step performed by the tool is to create a plan from the workflow (step 306). Each activity in the default path of the workflow corresponds to a task in the plan. The task identifies the scheduled start and stop times for the task. The tool then displays the plan (step 308). For example, the plan created from the workflow 400 depicted in Fig. 4 is shown in Fig. 7. The plan 700 includes two tasks 702 and 704 that correspond to the two activities 406 and 408 from the workflow 400. The first task 702 is scheduled to begin at 9 a.m. 706 on August 1, 2001 (not shown), and end at 6 p.m. 708 on the same day. The second

task 704 is scheduled to begin at 9 a.m. 710 on August 2, 2001 (712) and end at 5 p.m. 714 on the same day.

The plan 800 created from the workflow 500 depicted in Fig. 5 is shown in Fig. 8. The plan 800 includes two serial tasks 802 and 804 that correspond to the two serial activities 506 and 508 from the workflow 500. The plan 800 also includes two parallel tasks 806 and 808 that correspond to the two parallel activities 510 and 512 from the workflow 500. As shown in Fig. 8, “Serial 1” task 802 is scheduled to begin at 9 a.m. 810 on August 1, 2001 (812) and end at 5:30 p.m. 814 on the same day. The parallel tasks 806 and 808 are scheduled to start at the completion of the “Serial 1” task 802, and are scheduled to end at 6 p.m. 816 on August 2, 2001 (818). The “Serial 2” task 804 is scheduled to begin upon completion of the parallel tasks 806 and 808 and is scheduled to end at 6 p.m. 820 on August 3, 2001 (822).

The plan 900 created from the workflow 600 depicted in Fig. 6 is shown in Fig. 9. The plan 900 includes a task 902 corresponding to the activity “Get Parts” 606, followed by a task 904 corresponding to the activity “L or Rt Handed?” 608. The following task 906 corresponds to the activity “Right” 610. The final task 908 corresponds to the activity “Complete Assembly” 620. The plan 900 depicts the default path, and does not include any of the tasks corresponding to the non-default path. Although the start and end times are not depicted in the plan 900 shown in Fig. 9, each task has a scheduled start and stop time.

Returning to the high-level process of Fig. 3, the tool then activates the plan (step 310). Next, the tool manages the execution of the activated plan (step 312). The tool also modifies the display of the plan as each task is executed (step 314). The tool then determines whether the execution of the plan is complete (step 316). If the execution of the plan is complete, processing ends. Otherwise, processing continues to step 312.

For the exemplary plan 700 depicted in Fig. 7, upon activation, the first task 702 begins execution. The tool depicts the executing task 1002 by darkening the outer borders of the block representing the task 1002, as depicted in the plan 1000 shown in Fig. 10. After completion of the task, the tool depicts the executed task 1102 as a darkened block in the plan 1100, as shown in Fig. 11. At this point, the second task 1104 begins execution, as indicated by the darkened outer borders of the task 1104. Finally, after both tasks 1102 and 1104 of the plan 1100 have been executed, both tasks 1202 and 1204 are depicted as darkened blocks in the plan 1200, as shown in Fig. 12. In this embodiment, the tool represents an executing task with darkened borders and represents an executed task as a darkened block. One skilled in the art, however,

will recognize that any visible change in the blocks representing the tasks, e.g., a change in shape, color, shading, etc., may be used to represent the tasks in their various states. Thus, the representation of the tasks used in the methods, systems, and articles of manufacture consistent with the present invention are not limited to those used in the present embodiment.

5 The activation and execution of the tasks of the plan 800 depicted in Fig. 8 are shown in Figs. 13-16. Fig. 13 depicts the state of the plan 1300 while the “Serial 1” task 1302 is executing. Fig. 14 depicts the state of the plan 1400 after execution of the “Serial 1” task 1402, while the “Parallel 1” and the “Parallel 2” tasks 1404 and 1406 are executing. Fig. 15 depicts the state of the plan 1500 after execution of the “Serial 1” task 1502 and the “Parallel 1” and the
10 “Parallel 2” tasks 1504 and 1506, while the “Serial 2” task 1508 is executing. Finally, Fig. 16 depicts the state of the plan 1600 after execution of the tasks 1602, 1604, 1606, and 1608.

As discussed above, Fig. 9 represents a plan 900 created from a workflow 600 having a logic block 608. The activation and execution of the tasks of the plan 900 following the default path are shown in Figs. 17-21, while the activation and execution of the tasks of the plan 900
15 following the non-default path are shown in Figs. 22-27.

Fig. 17 depicts the state of the plan 1700 while the “Get Parts” task 1702 is executing. Fig. 18 depicts the state of the plan 1800 after the execution of the “Get Parts” task 1802, while the “L Or Rt Handed?” logic task 1804 is executing. Upon selection of the default path, the plan 1900 shown in Fig. 19 depicts both the “Get Parts” task 1902 and the “L Or Rt Handed?” logic task 1904 in executed states, while the “Right” task 1906 is depicted in an executing state. After
20 the execution of the “Right” task 1906 is complete, the state of the plan 2000 is depicted in Fig. 20 with the “Get Parts” task 2002, the “L Or Rt Handed?” logic task 2004, and the “Right” task 2006 in executed states and with the “Complete Assembly” task 2008 in an executing state. Finally, upon completion of the “Complete Assembly” task 2008, the state of the plan 2100 after
25 execution of the tasks 2102, 2104, 2106, and 2108 is complete is depicted in Fig. 21.

Alternatively, if the non-default path is to be chosen, the execution of the plan is initially the same as when the default path is chosen. Thus, as depicted in Fig. 22, the plan 2200 begins with the execution of the “Get Parts” task 2202. After completion of the “Get Parts” task 2202, the plan 2300 shown in Fig. 23 depicts the “Get Parts” task 2302 in an executed state while the
30 “L Or Rt Handed?” task 2304 is shown in an executing state. At this point, the resource assigned to choose the default or the non-default path chooses the non-default path, thus completing the execution of the “L Or Rt Handed?” task 2404, as indicated in Fig. 24. Upon

selection of the non-default path, the tool 200 modifies the plan 2400 to correspond to the non-default path of the corresponding workflow. The plan 2400 depicts the tasks included in the non-default path. Thus, the plan 2400 includes the “Left” and “Left Special” tasks 2406 and 2408 rather than the “Right” task 2306, which is depicted in Fig. 23 before the non-default path was chosen. As shown in Fig. 24, the “Left” task 2406 is executing. Fig. 25 depicts the plan 2500 after the “Get Parts” task 2502, the “L Or Rt Handed?” logic task 2504, and the “Left” task 2506 have been executed, while the “Left Special” task 2508 is executing. Continuing with the execution of the plan, Fig. 26 depicts the state of the plan 2600 after the “Get Parts” task 2602, the “L Or Rt Handed?” logic task 2604, the “Left” task 2606, and the “Left Special” task 2608 are done executing, while the “Complete Assembly” task 2610 is executing. Finally, Fig. 27 depicts the state of the plan 2700 after completion of the tasks 2702, 2704, 2706, 2708, and 2710.

Retrieving Or Creating A Workflow

Figs. 28A-C depict a flow diagram illustrating an exemplary process for retrieving or creating a workflow, i.e., step 302 in Fig. 3. Initially, the tool 200 determines whether to use an existing process or workflow group (step 2802). A workflow group is a collection of workflows (e.g., a directory or folder containing the collection of workflows) created by the Client Interface 134 on WebDAV Storage 142. Each workflow group is created by the Client Interface 134 on WebDAV Storage 142 with the “workflow group” property as explained further below. When creating a workflow, the Client Interface 134 allows the enterprise affiliate to store the workflow within an identified workflow group so that any enterprise affiliate using the Client Interface 134 is able to easily identify related workflows. For example, software-related workflows may be stored within the same workflow group so that an enterprise affiliate is able to quickly locate a desired workflow in order to create a corresponding plan using the Client Interface 134. One skilled in the art will appreciate that Client Interface 134 may store a workflow on WebDAV Storage 142 without associating the workflow with a workflow group.

The tool 200 receives user input from an enterprise affiliate with system administrative privileges or permissions, such as a project manager, to determine whether to retrieve an existing workflow group or to create a new workflow group. If the tool 200 determines that it will use an existing workflow group, the tool 200 receives an identification of the workflow group from the enterprise affiliate (step 2804). In one implementation, the Client Interface 134 may retrieve the

identifications for the workflow groups on the WebDAV Storage 142 by requesting that the folders or directories on WebDAV Storage 142 having a “workflow” property be returned by the WebDAV Server 140. The Client Interface may use any known method in accordance with WebDAV protocol to request that the WebDAV Server 140 return any directory or folder on WebDAV Storage 142 that corresponds to a workflow group. The tool 200 may then display the available workflow groups to allow the enterprise affiliate to select one of the available workflow groups. For example, as shown on the user interface 2900 depicted in Fig. 29, the tool 200 may display a hierarchical view 2902 of an identified workflow group 2904 stored on the root directory 2906 of WebDAV Storage 142. Alternatively, the enterprise affiliate may enter the identification of the desired workflow group to the tool 200 for retrieval. Using the identification, the tool 200 then retrieves the workflow group (step 2806).

If the tool 200 determines that a new workflow group will be created, the tool 200 receives the name of the workflow group from the enterprise affiliate (step 2808). For example, the enterprise affiliate may request a new workflow group by clicking on “Process Designer” button 2908 of the user interface 2900 depicted in Fig. 29. The enterprise affiliate may, alternatively, use any known data input technique, such as an icon or keyboard input, to indicate the request to the tool 200. Upon selecting the “Process Designer” button 2908, the tool 200 displays an exemplary user interface 3000 depicted in Fig. 30 for receiving a new workflow group identification 3002 via keyboard input from an enterprise affiliate using computer 102a or 102n.

After receiving the new workflow group identification, the tool 200 creates a new workflow group in storage (step 2810). For example, the tool 200 may create the workflow group on WebDAV Storage 142. To generate a new workflow group on WebDAV Storage 142, the Client Interface 134 sends the WebDAV Server 140 a request to create a new collection or folder on WebDAV Storage 142 with the same identification as the new workflow group identification 3002. In accordance with WebDAV protocol, the Client Interface 134 receives a response from the WebDAV Server 140 confirming that the new workflow group folder was created on WebDAV Storage 142. As previously discussed, when a new collection or folder is created using the WebDAV protocol, the WebDAV properties (e.g., “date of creation,” “property name” and “lockdiscovery” properties) are created and stored in association with the new directory by the WebDAV Server 140. Thus, when generating the new workflow group, the Client Interface 134 also sets the “property name” of the new workflow group to be

“workflow group” so that the Client Interface may subsequently use known WebDAV methods, such as “PropFind,” to retrieve the identification of each workflow group on WebDAV Storage 142.

After retrieving an existing workflow group or creating a new workflow group, the tool 200 determines whether to use an existing workflow (step 2812). The tool 200 receives user input from an enterprise affiliate with system administrative privileges or permissions to determine whether to retrieve an existing workflow or to create a new workflow. If the tool 200 determines that it will use an existing workflow, the tool 200 receives an identification of the workflow from the enterprise affiliate (step 2814). In one implementation, the Client Interface 134 may retrieve the identifications for the workflows in the selected workflow group and display the available workflows to allow the enterprise affiliate to select one of the available workflows. Alternatively, the enterprise affiliate may enter the identification of the desired workflow to the tool 200 for retrieval. Using the identification, the tool 200 then retrieves the workflow (step 2816).

If the tool 200 determines that a new workflow will be created, the tool 200 receives the name of the workflow from the enterprise affiliate (step 2818). For example, the enterprise affiliate may request a new workflow by clicking on the desired workflow group 3102 and selecting the “New Process” option 3104 from a pull-down menu 3106 on the user interface 3100 depicted in Fig. 31. The enterprise affiliate may, alternatively, use any known data input technique, such as an icon or keyboard input, to indicate the request to the tool 200. Upon selecting the “New Process” option 3104, the tool 200 may display the exemplary dialog box 3200 depicted in Fig. 32 to the enterprise affiliate. The enterprise affiliate may then enter the name of a new workflow 3202. After receiving the name for the workflow, the tool 200 creates the workflow in storage (step 2820).

Figs. 33A-C depict an exemplary workflow definition file 3300 that is produced by the tool 200 when the workflow 600 depicted in Fig. 6 is created. The name 3302 of the workflow, “Logic Branch Project,” is identified in the workflow definition file 3300. Also, as shown in the definition file 3300, the workflow 600 does not have a workflow group 3304. The element 3306 in the workflow definition file 3300 represents the “Get Parts” activity 606. Similarly, the element 3308 (Fig. 33C) represents the “L or Rt Handed?” logic activity 608, the element 3310 represents the “Right” activity 610, the element 3312 represents the “Left” activity 612, the element 3314 represents the “Left Special” activity 614, and the element 3316 represents the

“Complete Assembly” activity 620. The start element 602 is represented by the element 3318, and the end element 604 is represented by the element 3320.

The next step performed by the tool 200 is to receive an indication of the type of activity to be created for the workflow (step 2822 in Fig. 28B). As discussed above, the activity may be a standard activity or a logic activity. For example, the workflow 3402 depicted in the user interface 3400 of Fig. 34 includes five standard activities 3404, 3406, 3408, 3410, and 3412. The workflow 3402 also includes one logic activity 3414. The selection of the type of activity may be made by clicking on the icon for a standard activity 3416 or the icon for the logic activity 3418. Alternatively, any known data input technique, such as a pull-down menu or keyboard input, may be used to select the type of activity.

After receiving an indication of the type of activity, the tool 200 receives the name of the activity (step 2824). The names of the activities depicted in the workflow 3402 are included with the activity. Thus, the name of activity 3404 is “Assignment,” the name of activity 3406 is “Analysis,” etc.

Returning to the example workflow 600 depicted in Fig. 6, the name of the first activity 606 is “Get Parts,” which is identified by the element 3322 in the workflow definition file 3300 of Fig. 33. Similarly, the name of the logic activity 608 is “L or Rt Handed?,” which is identified by the element 3324. The name of the activity 610 is “Right,” as identified by the element 3326. The name of the activity 612 is “Left,” as identified by the element 3328. The name of the activity 614 is “Left Special,” as identified by the element 3330. Finally, the name of the activity 620 is “Complete Activity,” as identified by the element 3332.

After receiving a name for the activity, the tool 200 receives an indication of the role responsible for the activity (step 2826). As discussed above, the Client Interface (via Resource Manager Module 206) allows an enterprise affiliate to identify a role or role profile that may be assigned to an activity of the workflow. A role profile includes a Rolename that represents a “capability” or “skill set,” which is needed to perform a task of a plan created from the workflow, where the task corresponds to the activity of the workflow. For example, Fig. 35 depicts a user interface 3500 displayed by the Client Interface to receive a role profile. In the implementation shown in Fig. 35, the Client Interface receives a Rolename 3502 (e.g., “Project Manager”) for the role profile via the enterprise affiliate clicking on an “Add” button 3504 and then entering Rolename 3502 in a dialog box 3506 that is displayed by the Client Interface. In another implementation, the Client Interface may also receive as additional entries (not shown)

to dialog box 3506 a skill and an associated skill level for Rolename 3502 as part of this role profile. For example, the enterprise affiliate may indicate to the Client Interface via the additional entries to dialog box 3506 that the Rolename 3502 of "Project Manager" be associated with a skill entitled "Object-oriented software programming" and with a skill strength of "7" on a scale of 10. Assuming an enterprise affiliate is developing a workflow for producing a software development tool, the enterprise affiliate may assign to activities in the workflow the "Project Manager" role profile with this skill and skill level. Thus, when a plan is created from this workflow, a resource having the appropriate skill and skill level will automatically be assigned by the Client Interface to tasks corresponding to the activities with the "Project Manager" role assignment.

The tool 200 stores the role profiles in association with the selected workflow activity on WebDAV Storage 142. The tool 200 saves significant costs in developing multiple workflows by allowing the enterprise affiliate to store the role profiles in association with the selected workflow activity on WebDAV Storage 142 so that the role profiles may be available for the enterprise affiliate to assign to an activity of another workflow that is also related to the selected workflow activity. In one implementation, the Client Interface stores the role profiles in a single role definition file (not shown) on WebDAV Storage 142. In another implementation, the Client Interface stores the role profiles in separate files (not shown) on WebDAV Storage 142. Each separate file has a name that is the same as the received Rolename 3502. In this implementation, using known WebDAV protocol, the Client Interface defines an associated WebDAV property having a common name, such as "role profile," so that the Client Interface may later retrieve the role profiles stored on WebDAV storage.

The role profiles may also be stored with the workflow definition file. As shown in the workflow definition file 3300 depicted in Fig. 33, the role profile 3334 for the "Get Parts" activity 606 indicates that the role responsible for the activity is "Assembler" 3336. Similarly, the role profile 3338 for the "L or Rt Handed?" activity 608 indicates that the role responsible for the activity is "Assembler" 3340. The role profile 3342 for the "Right" activity 610 indicates that the role responsible for the activity is "Assembler" 3344. The role profile 3346 for the "Left" activity 612 indicates that the role responsible for the activity is "Assembler" 3348. The role profile 3350 for the "Left Special" activity 614 indicates that the role responsible for the activity is "Assembler" 3352. Finally, the role profile 3354 for the "Complete Assembly" activity 620 indicates that the role responsible for the activity is "Assembler" 3356.

The next step performed by the tool 200 is to determine whether the activity has any predecessor activities (step 2828). If the activity does have a predecessor activity, the tool 200 receives an indication of the predecessor activities from the workflow definition file (step 2830). After checking for any predecessor activities and/or receiving the predecessor activities, the tool 200 determines whether the activity has any successor activities (step 2832). If the activity has a successor activity, the tool 200 receives an indication of the successor activities from the workflow definition file (step 2834). In the user interface 3400 of Fig. 34, the “Path” icon 3420 is used to connect the predecessor activity to the successor activity. For example, in the workflow 3402, a path 3422 was drawn from the “Assignment” activity 3404 to the “Analysis” activity 3406. Thus, the “Assignment” activity 3404 is the predecessor activity to the “Analysis” activity 3406, and the “Analysis” activity 3406 is the successor activity to the “Assignment” activity 3404. Alternatively, a “Vertical For/Join” icon 3424 or a “Horizontal Fork/Join” activity may be used to connect more than one predecessor activities to a successor activity, or to connect a predecessor activity to more than one successor activities.

In the workflow 600 depicted in Fig. 6, the activity ID 3358 of the “Get Parts” activity 606 is “10.” The predecessor 3360 to the “Get Parts” activity 606 has an ID of “11” 3362, which corresponds to the start element 602. The successor 3364 to the “Get Parts” activity 606 has an ID of “1522” 3366, which corresponds to the “L or Rt Handed?” logic activity 608. The predecessor 3368 to the “L or Rt Handed?” logic activity 608 has an ID of “10” 3358, which corresponds to the “Get Parts” activity 606. Because the “L or Rt Handed?” activity 608 is a logic activity, it has both a default successor and a non-default successor. Thus, the workflow definition file 3300 identifies two paths out of the “L or Rt Handed?” logic activity 608, one path 3370 has an ID of “1525” 3372, which corresponds to the “Right” activity 610, and the other path 3374 has an ID of “1523” 3376, which corresponds to the “Left” activity 612. The element representing the “L or Rt Handed?” logic activity 608 also identifies that the default path 3378 has an ID of “1525” 3372, which corresponds to the “Right” activity 610. The predecessor 3380 to the “Right” activity 610 and the predecessor 3382 to the “Left” activity 612 have an ID of “1522” 3366, which corresponds to the “L or Rt Handed?” logic activity 608. The remaining predecessor and successors follow this convention.

After checking for any successor activities and/or receiving the successor activities, the tool 200 determines whether the activity has any on-entry scripts (step 2836). An on-entry script is a step to be performed by the tool 200 upon entry into the activity. For example, the on-entry

script may be to send an email notifying a user about the activity. Another example of an on-entry script is to retrieve the status of an artifact that may be necessary to complete the activity. The on-entry script may also be a request sent to an enterprise affiliate to obtain data. If the activity has an on-entry script, the tool 200 receives an indication of the on-entry scripts (step 2838). After checking for any on-entry scripts and/or receiving the on-entry scripts, the tool 200 determines whether the activity has any on-exit scripts (step 2840 in Fig. 28C). An on-exit script is a step to be performed by the tool 200 before exiting the activity. For example, the on-exit script may be to send an email notifying a user about the end of an activity. If the activity has an on-exit script, the tool 200 receives an indication of the on-exit scripts (step 2842). For example, the “Complete Assembly” activity 620 depicted in Fig. 6 includes both an on-entry script 3384 as well as an on-exit script 3386. Upon entering the task created from the “Complete Assembly” activity, the tool 200 sends an email to the owner indicating that the “Debugging period started” 3388. Prior to exiting the task created from the “Complete Assembly” activity, the tool 200 sends an email to the owner indicating that the “Debugging finished” 3390.

After checking for any on-exit scripts and/or receiving the on-exit scripts, the tool 200 determines whether the activity has any input conditions (step 2844). If the activity has an input condition, the tool 200 receives an indication of the input conditions (step 2846). After checking for any input conditions and/or receiving the input conditions, the tool 200 determines whether the activity has any output conditions (step 2848). If the activity has an output condition, the tool 200 receives an indication of the output conditions (step 2850). The output condition 3391 for the “Get Parts” activity 606 has an ID of “1527” 3392 (Fig. 33B), and is a document-type condition, as indicated by the “linkable1” identity 3393 in the element 3394 representing the condition 3391. In general, based on the condition 3391, the tool 200 (in particular, the Workflow Engine 222) monitors the state of an artifact for an activated “Get Parts” task created from the “Get Parts” activity 606 until the state of the artifact is the “INITIAL” state 3395 before the tool 200 continues with the next task in the plan. Similarly, the output condition 3396 for the “Right” activity 610 has an ID of “1533” 3397. The output condition 3396 for the “Right” activity 610 is also a document-type condition, as indicated by the “linkable1” identity 3398. This condition 3396 signals the tool 200 to monitor the state of an artifact until it is in the “RIGHT” state 3399.

Fig. 36 depicts an exemplary user interface 3600 displayed by the Client Interface 134 to include either a document-oriented 3602 or a script (or logic)-oriented 3604 condition. As shown in Fig. 36, the Client Interface 134 may receive the request to add a condition to the activity via a pull-down menu selection 3606. The enterprise affiliate may, however, use any known data input technique to request that a condition be added to an activity, such as an icon or keyboard input, to indicate the request to the Client Interface 134. If the enterprise affiliate selects a document-oriented condition, the enterprise affiliate may be presented with the user interface 3700 depicted in Fig. 37 to identify the properties of the condition to the Client Interface 134. The condition properties 3702 include condition-name property 3704 for the document-type condition model. In the example shown in Fig. 37, the Client Interface 134 receives the condition-name property 3704 via a keyboard input by the enterprise affiliate. The Client Interface 134 uses the condition-name property 3704 to distinguish the condition model to be created from other condition models stored on WebDAV Storage 142. The Client Interface 134 may store the document-type condition model file on WebDAV Storage 142 having the same name as the condition-name property 3704. In another implementation, the Client Interface 134 may store the condition-name property 3704 as a WebDAV property stored in association with the document-type condition model file on WebDAV Storage 142.

The Client Interface 134 also receives a link-parameter property 3706 as one of Condition properties 3702 for the document-type condition model to be created by the Client Interface. As shown in Fig. 37, the enterprise affiliate may identify link-parameter property 3706 to the Client Interface via keyboard input. Link-parameter property 3706 may be used by an enterprise affiliate in an activity-related script that is identified to the Client Interface during the creation of a workflow as described below. Thus, when executing the activity-related script in a task of a plan created from the workflow, the Workflow Engine 222 in Fig. 2 is able to locate the corresponding document condition upon so that a corresponding input or output condition may be evaluated by the Workflow Engine 222.

The Client Interface 134 may also receive a description property 3708 as one of Condition properties 3702 for the document-type condition model to be created by the Client Interface. When creating a workflow as described below, the Client Interface may display description property 3708 in association with condition-name property 3704 to allow an enterprise affiliate to effectively choose whether the document-type condition model should be assigned to an activity of the workflow.

The Client Interface may also receive one or more triggering-event properties 3710 for the document-type condition model. In the example shown in Fig. 37, the Client Interface may receive the triggering-event properties as one of the condition properties 3702 for the document-type condition model to be created by the Client Interface. Triggering-event properties 3710 indicate to the Workflow Engine 222 when to check the state property of a document condition as an entry or exit condition of an activated task. Triggering-event properties 3710 may include a "Write into document" event 3712, a "Change property of document" event 3714, a "Put document into repository" event 3716, a "copy or move into document" event 3718, and a "delete document" event 3720.

Next, the Client Interface 134 receives document state properties 3722 as one of the Condition properties 3702 for the document-type condition model to be created by the Client Interface. Document state properties 3722 identify possible values for a state property of a document condition that is created using the document-type condition model. As further explained herein, an enterprise affiliate who has been identified as the responsible owner of an activated task may change the state property of a document condition (e.g., from "DRAFT" to "APPROVED") using the Client Interface, which sends a request to WebDAV Server 140 in Fig. 2 to set the state property of the document condition as indicated by the enterprise affiliate. Workflow Engine 222 in Fig. 2 may then check the state property of the document condition on WebDAV Storage 142 when triggering-events 3710 occur.

The Client Interface also receives a location property 3724 as one of Condition properties 3702 identified by the enterprise affiliate for the document-type condition model. Location property 3724 is a unique identifier or URL for a document template that the Client Interface uses to create the document condition that is then stored by the Client Interface on WebDAV Storage 142. Location property 3724 may be a location on secondary storage device 116 of computer 102a or a location on WebDAV Storage 142. As described in greater detail below, the document condition is created by the Client Interface 134 when a plan is instantiated or created from a workflow having an activity with an entry or exit condition created using the document-type condition model. Finally, the Client Interface receives application property 3726 as one of Condition properties 3702 identified by the enterprise affiliate for the document-type condition model. Application property 3726 is a unique identifier or URL for an application, such as Microsoft Word, that the Client Interface may run to create an instant of the document template that may be found at the location specified by location property 3724. The Client

Interface uses the instant of the document template to create and store the document condition on WebDAV Storage 142.

Fig. 38 depicts an exemplary user interface 3800 displayed by the Client Interface 134 to receive the condition properties 3802 for a logic-type condition model that is to be created by the Client Interface 134. The condition properties 3802 include a condition-name property 3804 for the document-type condition model. In the example shown in Fig. 38, the Client Interface 134 receives the condition-name property 3804 via a keyboard input by the enterprise affiliate. The Client Interface 134 uses the condition-name property 3804 to distinguish the logic-type condition model to be created from other condition models stored on WebDAV Storage 142. As described below, the Client Interface 134 stores a logic-type condition model file on WebDAV Storage 142 that has the same name as condition-name property 3804. In another implementation, the Client Interface 134 may also store condition-name property 3804 as a WebDAV property stored in association with the logic-type condition model file on WebDAV Storage 142.

In the example shown in Fig. 38, the Client Interface 134 may receive a description property 3806 as one of the Condition properties 3802 for the logic-type condition model to be created by the Client Interface 134. When creating a workflow as described below, the Client Interface 134 may display the description property 3806 in association with the condition-name property 3804 to allow an enterprise affiliate to effectively choose whether the logic-type condition model should be assigned to an activity of the workflow.

The Client Interface 134 may also receive one or more triggering-event properties 3808 for the logic-type condition model as one of the condition properties 3802 for the logic-type condition model to be created by the Client Interface 134. Triggering-event properties 3808 indicate to the Workflow Engine 222 when to check an entry or exit condition of an activated task. Triggering-event properties 3808 include: an "Absolute time" event 3810, a "Period" event 3812, a "URL change" event 3814, a "Task change" event 3816, and "any http request" event 3818. "Absolute time" event 3810 identifies a trigger for a specific data and time from the start time of the activated task. "Period" event 3812 identifies a trigger for a specific unit of time, such as once every minute. "URL change" event 3814 identifies a trigger when the contents of the directory or folder located at the URL changes. "Task change" event 3816 identifies a trigger for any time the activated task definition file or associated property changes. For example, when an enterprise affiliate that is responsible for the task uses the Client Interface

134 to identify that the task is complete, the Client Interface 134 in response sends a request to the WebDAV Server 140 to set the status property of the activated task to "FINISHED." As part of the processing for managing an activated plan as described below, the Workflow Engine 222 will receive this request before the WebDAV Server 140 and interpret the request as an example of a "Task change" event 3816. Similarly, "Any http request" event 3818 indicates to the Workflow Engine 222 to check the entry or exit condition of the activated task when any request is received from the Client Interface 134 that pertains to the activated task. For example, the Client Interface 134 may send a request to the WebDAV Server 140 to retrieve the activated task file so that a status of the activated task can be viewed by an enterprise affiliate. Workflow Engine 222 will receive this request before the WebDAV Server 140 and interpret the request as an example of an "Any http request" event 3818.

The Client Interface 134 may also receive a script 3820 as one of the condition properties 3802 for the logic-type condition model to be created by the Client Interface 134. Script 3820 is executed by the Workflow Engine 222 when a triggering-event occurs that corresponds to one of the triggering-event properties 3808 selected by the enterprise user using the Client Interface 134. As shown in Fig. 38, Script 3820 may include a script parameter 3822, a script value 3824 for script parameter 3822, and script content 3826 that may use the script parameter 3822 initialized to the script value 3824. The enterprise affiliate may provide the script content 3826 to the Client Interface 134 via a Script Editor User Interface 3900 in Fig. 39. Script Editor User Interface 3900 is displayed by the Client Interface 134 when the enterprise affiliate actuates button 3828 on user interface 3800 shown in Fig. 38. Script content 3820 may contain any known application program interface (API) script method that would be recognizable by the target processor interpreter on computer 106, such as Java™ Virtual Machine 150 in Fig. 1.

After checking for any output conditions and/or receiving the output conditions, the tool 200 determines whether there are any more activities to add to the workflow (step 2852). If there are more activities, the process continues at step 2822 for the next activity. If there are no more activities to add to the workflow, the tool 200 receives an indication of the starting point for the workflow (step 2854). Next, the tool 200 receives an indication of the ending point for the workflow (step 2856) before the process ends.

Fig. 40 depicts an exemplary user interface 4000 displayed by the Client Interface 134 to receive the properties of an activity of a workflow. As depicted, the name 4002 of the activity (e.g., "Specs Development"), the duration 4004 of the activity (e.g., 1 unit) and the role 4006

responsible for the activity may be entered by the enterprise affiliate responsible for creating or modifying the workflow. In addition, the enterprise affiliate may enter an on-entry scrip 4008 as well as an on-exit script 4010. The properties of the activity also include the location 4012 of the subprocess defining the activity if the activity consists of a workflow.

5 Creating A Plan From A Workflow

Figs. 41A-B depict a flow diagram illustrating the process of creating a plan from a workflow, i.e., step 306 in Fig. 3. At this point, the enterprise affiliate has already selected the workflow that will be used to create the plan. Initially, the tool 200 receives an indication of the plan name (step 4102). In selecting the plan name, the Client Interface 134 allows the enterprise affiliate to store the project plan within an identified project plan group so that any enterprise affiliate using the Client Interface 134 is able to easily identify related project plans. A process plan group is a collection of project plans (e.g., a directory or folder containing the collection of project plans) created by the Client Interface 134 on WebDAV Storage 142. For example, the software-related project plans may be stored within the same project plan group so that an enterprise affiliate is able to quickly locate a desired project plan in order to create a corresponding plan using the Client Interface 134. One skilled in the art will appreciate that Client Interface 134 may store a project plan on WebDAV Storage 142 without associating the project plan with a project plan group. Fig. 42 depicts an exemplary user interface 4200 used to receive a project plan group.

In the implementation shown in Fig. 42, the Client Interface 134 receives a dialog box 4202 to enter the name of a new project plan group 4204 (e.g., "Software Projects") after clicking on a "Create Group" button 4206. Alternatively, if the enterprise affiliate decides to select an existing project plan group, the tool 200 provides the enterprise affiliate with a list 4300 of available project groups from which the enterprise affiliate may choose, as depicted in Fig. 43. The tool 200 then provides the enterprise affiliate with a dialog box 4400 to enter the name 4402 of the project, as shown in Fig. 44.

The next step performed by the tool 200 is to receive an indication of the working hours (step 4104). Fig. 45 depicts an exemplary timetable 4500 which the enterprise affiliate may use to identify the timetable defining a workday. As shown, the enterprise affiliate may select a timetable template 4502 with predefined working hours. The Standard Timetable 4504 includes five Working Days 4506 (Monday through Friday) and Working Hours 4508 from 8 a.m. (4510)

through 1 p.m. (4512) and from 2 p.m. (4514) until 5 p.m. (4516). Alternatively, the enterprise affiliate may select a 24 Hour Timetable 4518 or an Intensive Timetable 4520, i.e., more than the Standard Timetable 4504, but less than the 24 Hour Timetable 4518. The tool 200 also receives an indication of the start date and time for the project plan (step 4106). An exemplary dialog box 4600 may be used to select the start date and time 4602 and end date and time 4604.

The tool 200 then retrieves an activity from the workflow (step 4108). The tool 200 sets the start time of the task equal to the start date and time of the project plan (step 4110). Next, the tool 200 sets the end time of the task based on the start time of the task, the duration of the activity from which the task is based, and on the working hours (step 4112 in Fig. 41B). The tool 200 then receives an indication of the resource assigned to the task (step 4114).

For example, Fig. 47 depicts an exemplary workflow definition file 4700 that is produced by the tool 200 when the workflow 500 depicted in Fig. 5 is created. Fig. 48 depicts an exemplary project plan definition file 4800 created from the workflow definition file 4700. The element 4702 in the workflow definition file 4700 represents the “Serial 1” activity 506. As shown, the “Serial 1” activity 506 has a duration 4704 of 9 hours. If the working hours are determined based on the “24 Hour Timetable” 4818 and the start date and time for the project plan is 9 a.m. on August 1, 2001, the start time 4804 for the “Serial 1” task 4802 is 9 a.m. on August 1, 2001. The end time 4806 of the task 4802 occurs 9 hours later, i.e., at 6 p.m. on August 1, 2001.

Fig. 49 depicts an exemplary user interface 4900 displayed by the Client Interface 134 to assign users or resources to roles. The tool 200 displays a list of available users or resources 4902 and a list of the roles 4904 in a given workflow. In this embodiment, the enterprise affiliate is allowed to selectively add or remove available resources to a role by highlighting the resource and selecting either the “Add” button 4906 or the “Remove” button 4908, respectively. Alternatively, the enterprise affiliate may add or remove the resources to a role by selecting the “Add all” button 4910 or the “Remove all” 4912 button, respectively. Thus, the enterprise affiliate may identify to the tool 200 resources that are capable of performing the role when assigned to a task in the plan. As discussed below, the tool 200 may automatically assign a resource to a role of a task in the plan based on the identified, capable resources for the role.

The properties of an activity may be modified using the exemplary user interface 5000 depicted in Fig. 50. The user interface 5000 displays the name 5002 of the activity, the duration 5004 assigned to the corresponding activity, the start date and time 5006 for the activity, the end

date and time 5008 for the activity, the responsible role 5010 assigned to the corresponding activity, the responsible resource or user 5012 assigned to the task, the owners 5014 of the task, the priority 5016 of the task, the on-entry script 5018 of the task, and the on-exit script 5020 of the task. The responsible resource 5012 of the task is the resource with the authority to notify the tool 200 when the task is complete. The owner(s) 5014 of the task, on the other hand, are notified when the task is started or completed, but do not have the authority to modify the tool 200 when the task is complete.

The next step performed by the tool 200 is to determine whether there are any more activities in the workflow (step 4116). If there are no more activities, the process ends. If there are more activities, the tool 200 retrieves the next activity (step 4118). The tool 200 then sets the start time of the task equal to the end time of the predecessor task (step 4120). The process then continues at step 4112.

The next activities that are retrieved by the tool 200 are "Parallel 1" 510 and "Parallel 2" 512. Element 4706 and element 4708 in the workflow definition file 4700 represent these activities 510 and 512. The durations 4710 and 4712 of both of these activities is 24 hours. The start time 4812 and 4814 of these tasks 4808 and 4810 is equal to the end time 4806 of the predecessor task, i.e., 6 p.m. on August 1, 2001. Because the duration 4710 and 4712 of the activities 510 and 512 is 24 hours, the end times 4816 and 4818 of these tasks 4808 and 4810 occur 24 hours later, i.e., at 6 p.m. on August 2, 2001. The next activity retrieved by the tool 200 is "Serial 2" 508. The element 4714 in the workflow definition file 4700 represents this activity. The duration 4716 of the "Serial 2" activity 508 is 24 hours. The start time 4822 of the task 4820 created from the "Serial 2" activity 508 is the end time 4816 and 4818 of the predecessor task, i.e., 6 p.m. on August 2, 2001. Because the duration 4716 of the "Serial 1" activity is 24 hours, the end time 4824 of the task 4820 is 6 p.m. on August 3, 2001. The project plan is displayed in the Gantt chart 5100 depicted in Fig. 51. As shown, the "Serial 1" task 5102 is scheduled to execute from 9 a.m. 5104 on August 1, 2001 (5106) through 6 p.m. 5108 on the same day. The "Parallel 1" task 5110 and the "Parallel 2" task 5112 are scheduled to execute from 6 p.m. 5108 on August 1, 2001 (5106) through 6 p.m. 5114 on August 2, 2001 (5116). Finally, the "Serial 1" task 5118 is scheduled to execute from 6 p.m. 5114 on August 2, 2001 (5116) through 6 p.m. 5120 on August 3, 2001 (5122). Note that an enterprise affiliate using the Client Interface 134 on the computer 102a may create a plan from the workflow 600 at the same

time that a second enterprise affiliate using the Client Interface 134 on computer 102n creates a second plan from the workflow 600.

After the project plan is created from the workflow, the plan may be activated. As depicted in Fig. 52, the enterprise affiliate may activate the project by selecting the “Activate Project” option 5202 from the pull-down menu 5200. The enterprise affiliate may, however, use any known data input technique, such as an icon or keyboard input, to indicate the request to Client Interface 134.

In one implementation, the Client Interface 134 then sends an activate request to the WebDAV server 140 to change the status of the plan definition file to “Active.” As discussed further below, the Workflow Engine 222 may intercept this request and process the request in preparation for managing the execution of the activated plan. Once the plan is created and stored on WebDAV storage 142, any enterprise affiliate with access privileges may activate the plan using the Client Interface 134 from any computer 102a and 102n.

Adding A Resource

Fig. 53 depicts a flow diagram illustrating an exemplary process performed by the Client Interface 134 to add a new resource to the list of available resources. The Client Interface 134 may later assign the resource to a plan in accordance with methods and systems consistent with the present invention. Initially, the Client Interface 134 receives a request to add a new resource (step 5302). As shown in Fig. 54, the Client Interface 134 may receive the request to add a new resource via a pull-down menu selection 5402 and 5404 that is chosen by an enterprise affiliate. The enterprise affiliate may, however, use any known data input technique, such as an icon or keyboard input, to indicate the request to the Client Interface 134.

Next, the Client Interface 134 determines whether the request is to import the resource information (step 5304). In the implementation shown in Fig. 54, an enterprise affiliate requests that the Client Interface 134 import a resource profile containing the resource information by choosing the pull-down menu selection 5404. Alternatively, the enterprise affiliate may request that the Client Interface 134 create the resource profile from resource information that the enterprise affiliate provides to the Client Interface 134. Thus, if the request is not to import the resource information, the Client Interface 134 receives the resource information from the enterprise affiliate (step 5306). As shown in Fig. 54, the Client Interface 134 may receive resource information 5404 for an enterprise affiliate (e.g., a user or person) that may later be

assigned to a plan by the Client Interface 134 in accordance with processes described in greater detail below. The Resource Information 5404 may include a login name 5408, a resource name 5410 that the Client Interface 134 is to use when assigning the resource to a task of a plan, and an e-mail address 5412 that the Client Interface 134 or the Workflow Engine 222 may use to notify the resource of an assignment or another event.

The Client Interface 134 may also receive other resource information (not shown) for other types of resources (e.g., equipment, facilities, computer systems, or other known entities) that may be assigned to any task of a plan. The other resource information may include: a resource name that the Client Interface 134 is to use when assigning the resource to a task of a plan; a resource owner name that identifies a manager or other enterprise affiliate who is responsible for the named resource; and an e-mail address for the named resource owner, which the Client Interface 134 or the Workflow Engine 222 may notify when the named resource is assigned to a task or for another associated event.

Resource information 5404 may also include one or more skill identifiers that indicate one or more capabilities that a task of a plan may require for the task to be completed. Skill identifiers may include any foreseeable skill for the named resource, including a user, equipment, facilities, computer systems, or other known entities that may be assigned to any task of a plan. For example, when the named resource is an enterprise affiliate, the skill identifiers that may be identified for the enterprise affiliate may include: "Java programming," "architecture," or "carpentry." When the named resource is equipment, the skill identifiers may include "punch-press" or "Windows NT Operating System." Resource information 5404 may also include a skill strength (not shown) for each skill identifier. The skill strength may be used by the tool to differentiate one resource from another resource when matching a resource to a role of a task in a plan.

Resource information 5404 may also include an availability timetable (not shown) that indicates to the Client Interface 134 the calendar days, the hours in a weekday, and the hours in a weekend day that the named resource is available to work. Resource information 5404 may also include an assignment timetable (not shown) that has assigned calendar days. The assigned calendar days indicate to the Client Interface 134 which calendar days the named resource has been assigned to one or more tasks. In addition, the assignment timetable may include unique identifiers or URLs for the one or more tasks to which the named resource has been assigned. Thus, the Client Interface 134 or the Workflow Engine 222 may access the one or more tasks

that the named resource has been assigned when performing processing for resource leveling of a plan in accordance with methods and systems consistent with the present invention.

If the request is to import the resource information, the Client Interface 134 receives access information for a "Lightweight Directory Access Protocol (LDAP)" resource directory entry (e.g., a resource profile) on the network 108 of Fig. 1 (step 5308). Fig. 55 depicts an exemplary user interface 5500 showing access information 5502 received by the Client Interface 134. Access information 5502 includes an LDAP Server 5504 (e.g., "Frodo") on the network 108, an LDAP Port 5506 for the Client Interface 134 to communicate with the LDAP Server 5504, and a resource distinguished name (DN) 5508 identifying the location on LDAP Server 5504 where the resource profile may be found. The access information 5502 may be default access information that the Client Interface 134 retrieves from a configuration file (not shown) on the computer 102a, or it may be access information entered by an enterprise affiliate. In the implementation illustrated in Fig. 55, the access information 5502 may also include: a security distinguished name (DN) 5510, a password 5512, and a login alias 5514. Security DN 5510 identifies to the Client Interface 134 where a security profile for the enterprise affiliate is located. The Client Interface 134 uses the password 5512 and the login alias 5514 to access the resource information on the LDAP Server 5504 in accordance with privileges identified in the security profile.

Having received the access information for the LDAP directory entry on network 108, the Client Interface 134 retrieves the resource information using the LDAP access information (step 5310). The resource information that the Client Interface 134 retrieves includes resource profiles for a user, equipment, facilities, computer systems, or other known entities that may be assigned to any task of a plan.

After the resource information is received from the enterprise affiliate or is retrieved using LDAP access information, the Client Interface 134 stores the resource information in resource profiles on the WebDAV Storage 142 (step 5312).

Fig. 56 depicts an exemplary resource file 5600 that the Client Interface 134 may use to store resource profiles 5602, 5604, 5606, and 5608 on WebDAV Storage 142. As shown in Fig. 56, the resource profile 5600 includes a unique identifier or URL 5612 where the resource profile 5600 is to be stored on the WebDAV Storage 142. Each resource profile 5602, 5604, 5606, and 5608 may be stored separately by the Client Interface 134 on WebDAV Storage 142. In the implementation shown in Fig. 56, the resource profile 5602 includes resource information

5610 that corresponds to an enterprise affiliate that may be assigned to a task of a plan. In another implementation, the resource information 5610 may be added as properties rather than as the content of the resource profile 5602 on WebDAV Storage 142. This implementation may be advantageous as the Client Interface 134 or the Workflow Engine 222 may use a known WebDAV method to retrieve resource profiles from the WebDAV Storage 142 that have the same property. For example, the WebDAV “PropFind” method may be used by the Client Interface 134 or the Workflow Engine 222 to retrieve the resource profiles having a skill identifier of “Java Programming” so that an available resource having this skill can be assigned to a task in accordance with processes described below.

10 Managing A Plan

Fig. 57 depicts a flow diagram illustrating an exemplary process performed by the Workflow Engine 222 to manage the execution of an activated plan. The Workflow Engine 222 may execute the process in Fig. 57 for each activated plan stored on WebDAV Storage 142. Thus, the tool manages the execution of multiple plans simultaneously.

Initially, the tool 200 waits until the current time and date are later than the start time and date (step 5702). Note that the Workflow Engine 222 may first retrieve the plan from WebDAV storage. At this point, the tool 200 selects a task from the activated project plan created from a workflow (step 5704). Next, the tool 200 determines whether there is an input condition (step 5706). If there is an input condition, the tool 200 waits until the input condition is met (step 5708). After the input condition is met or if there is no input condition, the tool 200 stores the actual start time (step 5710). The next step performed by the tool 200 is to determine whether there is an on-entry script to execute, such as a message to send to the resource (step 5712 in Fig. 57B). If there is an on-entry script, the tool 200 performs the on-entry script (step 5714). After performing the on-entry script or if there is no on-entry script, the tool 200 determines whether there is an output condition (step 5716). If there is an output condition, the tool 200 waits until the output condition is met (step 5718). After the output condition is met or if there is no output condition, the tool 200 determines whether there is an on-exit script (step 5720). If there is an on-exit script, the tool 200 performs the on-exit script (step 5722). After performing the on-exit script or if there is no on-exit script, the tool 200 stores the actual end time (step 5724). Then the tool 200 determines whether there are any more tasks in the project plan (step

5726). If there are no more tasks, the process ends. Otherwise, the process returns to step 5704 and selects the next task.

The plan 5800 created from the workflow 500 depicted in Fig. 5 is shown in Fig. 58. As shown in Fig. 58, “Serial 1” task 5802 is scheduled to begin at 9 a.m. 5804 on August 1, 2001 (5806) and end at 6 p.m. 5808 on the same day. The parallel tasks 5810 and 5812 are scheduled to start at the completion of the “Serial 1” task 5808, and are scheduled to end at 6 p.m. 5814 on August 2, 2001 (5816). The “Serial 2” task 5818 is scheduled to begin upon completion of the parallel tasks 5814 and is scheduled to end at 6 p.m. 5820 on August 3, 2001 (5822). Fig. 59 depicts an exemplary project plan definition file 5900 corresponding to the plan 5800 of Fig. 58.

Upon activation, the “Serial 1” task 6002 begins execution, as depicted by the task 6004 in the Gantt chart 6000 of Fig. 60. Contrary to the plan, however, the “Serial 1” task ends earlier than planned. As depicted in Fig. 61, the actual properties 6100 of the “Serial 1” task 6102 include the actual-start-date 6104 (i.e., year-2001 month-8 day-1 hour-9) and actual-finish-date 6106 (i.e., year-2001 month-8 day-1 hour-14, i.e., 2 p.m.). The actual execution 6204 of the “Serial 1” task 6202 is shown in the Gantt chart 6200 of Fig. 62.

Because the “Serial 1” task 6202 ended earlier than planned, both the “Parallel 1” task 6206 and the “Parallel 2” task 6208 begin execution at 2 p.m. 6210 rather than waiting until their scheduled start time of 6 p.m. The earlier execution 6212 and 6214 of these tasks 6206 and 6208 is also depicted in the Gantt chart 6200. As depicted in Fig. 63, the actual properties 6300 of the “Parallel 1” task 6302 and the “Parallel 2” task 6304 include the actual-start-date 6306 (i.e., year-2001 month-8 day-1 hour-14) and actual-finish-date 6308 (i.e., year-2001 month-8 day-2 hour-0). The actual execution 6406 and 6408 of the “Parallel 1” task 6402 and the “Parallel 2” task 6404 is shown in the Gantt chart 6400 of Fig. 64. The Gantt chart 6400 also visually indicates that the start time 6410 for the tasks 6402 and 6404 was 2 p.m. on August 1, 2001, while the end time 6412 for the tasks 6402 and 6404 was 12 a.m. on August 2, 2001.

Finally, the execution of the “Serial 2” task 6414 begins at 12 a.m. on August 2, 2001 (6412). As depicted in Fig. 65, the actual properties 6500 of the “Serial 2” task 6502 includes the actual-start-date 6504 (i.e., year-2001 month-8 day-2 hour-0) and actual-finish-date 6506 (i.e., year-2001 month-8 day-2 hour-12). The actual execution 6604 of the “Serial 1” task 6602, the actual execution 6608 of the “Parallel 1” task 6606, the actual execution 6612 of the “Parallel 2” task 6610, and the actual execution 6616 of the “Serial 2” task 6614, are shown in the Gantt chart 6600 of Fig. 66.

Improving A Resource Profile Based On Data Mined From Plans Created From A Workflow

In addition to the functionality described above, the integrated process modeling and project planning tool Client Interface saves an enterprise significant development costs and increases its operating efficiency by improving profiles of resources associated with a workflow based on data mined from plans created from the workflow. By improving the profiles of the resources, the Client Interface is able to optimize the automatic assignment or allocation of the resources to tasks of a plan when the plan is created from the workflow. The data mined by the Client Interface includes a task found in each plan that has an auto-assigned resource and has a replacement resource. The auto-assigned resource is a resource that the Client Interface automatically assigned to handle a role of the task when the Client Interface created the task from an activity of the workflow. The replacement resource is another resource that was manually assigned by an enterprise affiliate to replace the auto-assigned resource in handling the role of the task. The Client Interface improves a profile associated with the auto-assigned resource by removing a skill or decreasing a strength of the skill in the profile, where the skill is associated with the role of the mined task. The Client Interface improves a profile associated with the replacement resource by adding a skill or increasing a strength of the skill in the profile, where the skill is also associated with the role of the mined task. Thus, by improving the profiles for the auto-assigned resource and the replacement resource, the Client Interface is able to more effectively automatically allocate resources to plans that are subsequently created from the workflow by the enterprise affiliate, saving the enterprise affiliate time and effort needed to manually override any resources that were not optimally automatically assigned.

Figs. 67A through 67E depict a flowchart illustrating an exemplary process performed by the Client Interface to improve a profile of any resource that has been previously assigned to a role of a task in a plan created from a workflow. Initially, the Client Interface receives an indication to improve a profile of any resource associated with any role of a workflow (Step 6702). In one implementation, the Client Interface first receives an indication to improve a profile of any resource associated with any role of a workflow when prompted by an enterprise affiliate to optimize the allocation of resources to plans created from the workflow. The enterprise affiliate may prompt the Client Interface to optimize resource allocation via any known data input technique, such as clicking a mouse on a button of a user interface (not shown) displayed by the Client Interface. In this implementation, the Client Interface may respond by retrieving and displaying the identifications of all the workflow definition files stored on

WebDAV Storage 142 of Fig. 2. The enterprise affiliate may then identify one of the displayed workflow definition files to the Client Interface by clicking the mouse on the displayed identification corresponding to the one workflow definition file.

For clarification in the discussion to follow, when performing step 6702 of Fig. 67A, it is assumed that the enterprise affiliate has prompted the Client Interface to optimize resource allocation for plans created from the workflow identified as "AssemblyWorkflow." Fig. 68 depicts an excerpt of an exemplary workflow definition file 6800 for the identified "AssemblyWorkflow" 6802 as created and stored by the Client Interface in response to performing the process depicted in Fig. 3. Fig. 69 depicts an exemplary graphical representation of workflow definition file 6800 as displayed by the Client Interface when performing the process depicted in Fig. 3 to create the workflow or after retrieving the workflow from WebDAV Storage 142 in response to a request by enterprise affiliate to view the workflow. "AssemblyWorkflow" may have been designed by an enterprise affiliate using the Client Interface to define a workflow for developing a video game controller or a baseball mitt where both a left- or right-handed product may be assembled using a plan created from the workflow.

As shown in Figs. 68 and 69, the workflow includes a "Get Parts" activity (6804 in Fig. 68 and graphically depicted as 6904 in Fig. 69), a "Get Fasteners" activity (6806 and 6906), a "L Or Rt Handed?" logic activity (6808 and 6908), a "Left" activity (6910 and 6912), a "Left Special" activity (6812 and 6912), a "Right" activity (6814 and 6914), and a "Complete Assembly" activity (6816 and 6916). "L Or Rt Handed?" logic activity 6808 has an outgoing default-path 6838 (graphically depicted as a solid line 6918 in Fig. 69) that includes "Left" activity 6810 and "Left Special" activity 6812. "L Or Rt Handed?" logic activity 6808 has an outgoing non-default-path (graphically depicted as a dashed line 6920 in Fig. 69) that includes "Right" activity 6814.

Each activity 6804, 6806, 6808, 6810, 6812, 6814, and 6816 has a corresponding responsible role 6818, 6820, 6822, 6824, 6826, 6828, and 6830 that identifies a role assigned by the Client Interface as specified by an enterprise affiliate in accordance with methods and systems consistent with the present invention. As shown in Fig. 68, the Client Interface has assigned a role identified as "Gopher" 6832 to responsible roles 6818 and 6820. The Client Interface has assigned a role identified as "Assembler" 6834 to responsible roles 6822, 6824, 6826, and 6828. The Client Interface has also assigned a role identified as "Master Assembler" 6836 to responsible role 6830.

When the Client Interface creates a plan from workflow definition file 6800, the Client Interface creates tasks for the plan from activities 6804, 6806, 6808, 6810, 6812, and 6816 that are included in the default-path for workflow definition file 6800. The Client Interface also automatically assigns a resource to a role for each task based on a resource allocation process, such as matching a role having a skill and a corresponding skill strength to a resource having the same skill and the same or greater skill strength. Thus, when performing the resource allocation process, the Client Interface examines whether a resource has one or more skills that indicate one or more capabilities that a task of a plan may require for the task to be completed. As discussed in greater detail below, to perform the resource allocation process, the Client Interface accesses a profile for the role to identify the skill and the corresponding skill strength for the role (i.e., capabilities that a task of a plan may require) and accesses a resource profile for the resource to identify if the resource has the same skill and the same or greater skill strength (i.e., same or greater capabilities than the task of the plan requires).

Continuing with Fig. 67A, after receiving an indication to improve a profile of any resource associated with any role of a workflow, the Client Interface identifies each resource associated with each role of the workflow (Step 6703). In one implementation, the Client Interface identifies each resource associated with each role of the workflow identified as “AssemblyWorkflow” by accessing a workflow roles file 7000, shown in Fig. 70, that is stored on WebDAV Storage 142. To access workflow roles file 7000, the Client Interface associates an identification 6802 for “AssemblyWorkflow” definition file 6800 to an identification 7002 of workflow roles file 7000. In another implementation, workflow roles file 7000 has a link 7004 to “AssemblyWorkflow” definition file 6800 to allow the Client Interface to associate workflow roles file 7000 with “AssemblyWorkflow” definition file 6800.

Workflow roles file 7000 has role profiles 7006, 7008, and 7010 for “Gopher” role 7012, “Assembler” role 7014, and “Master Assembler” role 7016. The Client Interface is able to identify that “Gopher” role 7012, “Assembler” role 7014, and “Master Assembler” role 7016 correspond to roles 6832 in Fig. 68, 6834, and 6836 associated with “AssemblyWorkflow” definition file 6800. Each role profile 7006, 7008, and 7010 identifies a set 7018, 7020, and 7022 of skills (or capabilities) and corresponding skill strengths that have been previously defined by an enterprise affiliate using the Client Interface. Each role profile 7006, 7008, and 7010 also has a corresponding list 7024, 7026, and 7028 of capable resources. The capable resources in list 7024, 7026, and 7028 may have been previously identified by an enterprise

affiliate using the Client Interface or may have been previously identified by the Client Interface based on the Client Interface matching a skill and skill strength set for the role to skills and corresponding skill strengths for the capable resource. In another implementation, the Client Interface may allow an enterprise affiliate to indicate that “Gopher” role 6832 in “Get Parts” activity 6804 have a skill or a skill strength that is different from “Gopher” role profile 7006 in Fig. 70 or that is different from “Gopher” role 6832 of “Get Fasteners” activity 6806 as shown below:

```
<activity id="10" name="Get Parts" responsiblerole="Gopher" skillref="1"
strength="8" skillref="2" strength="5" optimize="yes" minplans="5" max-percentage="10">
```

*

*

```
<activity id="55" name="Get Fasteners" responsiblerole="Gopher" skillref="2"
strength="6" optimize="yes" minplans="5" max-percentage="10">
```

When the Client Interface identifies that the skill or the skill strength for “Gopher” role 6832 in “Get Parts” activity 6804 is different from skill set 7018 in “Gopher” role profile 7006 in Fig. 70, the Client Interface uses the different skill or the different skill strength for “Gopher” role 6832 in lieu of skill set 7018 to perform methods consistent with the present invention.

For example, Fig. 71 depicts an exemplary resource file 7100 created by the Client Interface to store resource profiles 7102, 7104, and 7106 that have been defined by an enterprise affiliate using the Client Interface. Each resource profile 7102, 7104, and 7106 identifies a resource 7108 (“Mister Assembler”), 7110 (“Mister Gopher”), and 7112 (“Mister Build”) and a set 7114, 7116, and 7118 of skills (or capabilities) and corresponding skill strengths for the identified resource. Each resource profile 7102, 7104, and 7106 also has a unique identifier or link 7120, 7122, and 7124 that allows the Client Interface to distinguish each resource from all other resources. In one implementation, the Client Interface stores resource file 7100 on WebDAV Storage 142. In another implementation, the Client Interface stores resource profiles 7102, 7104, and 7106 as separate files on WebDAV Storage 142.

Fig. 72 depicts an exemplary skills file 7200 created by the Client Interface to store skills 7202, 7204, 7206, 7208, 7210, 7212, and 7214 that may be assigned to a role or a resource by the Client Interface. Skills 7202, 7204, 7206, 7208, 7210, 7212, and 7214 have been defined by an enterprise affiliate using the Client Interface. Each skill set 7018 in Fig. 70, 7020, and 7022 identified in role profiles 7006, 7008, and 7010 and each skill set 7114 in Fig. 71, 7116, and

7118 identified in resource profiles 7102, 7104, and 7106 contain one or more of skills 7202, 7204, 7206, 7208, 7210, 7212, and 7214. For example, the Client Interface is able to identify that skill set 7018 for “Gopher” role profile 7006 has a skill 7028 that corresponds to “enterprise procurement logistics” skill 7202 and has a skill 7030 that corresponds to “Chicago plant receiving logistics” skill 7204 in skills file 7200. The Client Interface is also able to identify that skill set 7020 for “Assembler” role profile 7008 has a skill 7032 that corresponds to “electrostatic discharge training” skill 7206, a skill 7034 that corresponds to “electrical schematic reading” skill 7208, and a skill 7036 that corresponds to “soldering” skill 7210 in skills file 7200. In addition, the Client Interface is able to identify that skill set 7114 for “Mister Assembler” resource profile 7102 has skills 7126, 7128, 7130, 7132, and 7134 that correspond to “enterprise procurement logistics” skill 7202, “Chicago plant receiving logistics” skill 7204, “electrostatic discharge training” skill 7206, “electrical schematic reading” skill 7209, and “soldering” skill 7210 in skills file 7200. Moreover, the Client Interface is able to identify that skills 7126 and 7128 for “Mister Assembler” resource profile 7102 have corresponding skill strengths that match or exceed skill strengths corresponding to skills 7028 and 7030 for “Gopher” role profile. The Client Interface is also able to identify that skills 7130, 7132, and 7134 for “Mister Assembler” resource profile 7102 have corresponding skill strengths that match or exceed skill strengths corresponding to skills 7032, 7034, and 7036 for “Assembler” role profile. Thus, when performing resource allocation processing using methods and systems consistent with the present invention, the Client Interface may select “Mister Assembler” resource 7108 to be associated with or handle both “Gopher” role 7012 and “Assembler” role 7014 in a plan created from “AssemblyWorkflow” definition file 6800 in Fig. 68.

To identify each resource associated with each role of “AssemblyWorkflow,” the Client Interface first identifies that capable resource list 7024 in Fig. 70 associated with “Gopher” role 7012 has unique identifiers 7038 and 7040 that corresponds to “Mister Assembler” resource 7108 in Fig. 71 and to “Mister Gopher” resource 7110, respectively. The Client Interface further identifies that capable resource list 7026 in Fig. 70 associated with “Assembler” role 7014 has unique identifiers 7038 and 7042 that correspond to “Mister Assembler” resource 7108 in Fig. 71 and to “Mister Build” resource 7112, respectively. In addition, the Client Interface identifies that capable resource list 7028 in Fig. 70 for “Master Assembler” has a unique identifier 7042 that corresponds to “Mister Build” resource 7112.

After identifying each resource associated with each role of the workflow, the Client Interface receives a maximum percentage of manual assignments for each resource associated with each role of the workflow (Step 6704). Next, the Client Interface receives a maximum percentage of override assignments for each resource (Step 6706). Each maximum percentage of manual assignments reflects the number of times that the Client Interface should find the resource has been manually assigned to a role of a task within the number of plans that are mined by the Client Interface before the Client Interface notifies the enterprise affiliate of a suggestion to improve the resource. Each maximum percentage of override assignments reflects the number of times that the Client Interface should find the resource was automatically assigned by the Client Interface to a role of a task and then replaced by another manually assigned resource within the number of plans that are mined by the Client Interface before the Client Interface notifies the enterprise affiliate of a suggestion to improve the resource. Thus, each maximum percentage of manual assignments and each maximum percentage of override assignments for each resource represents a threshold before the Client Interface modifies the skill or the skill strength of the resource to improve automatic resource allocation by the Client Interface.

In one implementation, the enterprise affiliate identifies to the Client Interface each maximum percentage of manual assignments and each maximum percentage of override assignments for each resource via the same user interface (not shown) that the enterprise affiliate used to initiate the request to optimize the resource allocation. In another implementation, the enterprise affiliate identifies to the Client Interface each maximum percentage of manual assignments and each maximum percentage of override assignments for each resource for each resource via a user interface (not shown) displayed by the Client Interface for receiving resource identification, resource skills, and corresponding skill strengths. In this implementation, the Client Interface stores each maximum percentage of manual assignments and each maximum percentage of override assignments in the resource profile for each resource. In the example shown in Fig. 71, the Client Interface has received and stored a value of twenty percent (20%) for maximum percentage of manual assignments 7136, 7140, and 7144 associated with role profiles 7102, 7104, and 7106, respectively. The Client Interface has received and stored a value of twenty percent (20%) for maximum percentage of override assignments 7138, 7142, and 7146 associated with role profiles 7102, 7104, and 7106, respectively.

After receiving a maximum percentage of override assignments and a maximum percentage of manual assignments for each resource, the Client Interface sets an auto-assignment override counter for each resource associated to zero (Step 6708). The Client Interface uses the auto-assignment override counter for each resource to track how many times the Client Interface identified that the resource had been replaced by another resource that was manually assigned by an enterprise affiliate. The Client Interface also sets a manual-assignment counter for each resource to zero (Step 6710). The Client Interface uses the manual-assignment counter for each resource to track how many times the Client Interface identified that the resource had been manually assigned by an enterprise affiliate. Continuing with the above example, the Client Interface sets an auto-assignment override counter and a manual-assignment counter for “Mister Assembler” resource 7108 in Fig. 71, for “Mister Gopher” resource 7110, and for “Mister Build” resource 7112.

Next, the Client Interface sets a plan counter to zero (Step 6712). The Client Interface uses the plan counter to track the number of plans checked.

After setting a plan counter to zero, the Client Interface retrieves a plan created from the workflow (Step 6714). To retrieve the plan created from the workflow, the Client Interface uses known WebDAV protocol methods to parse each plan definition file stored on WebDAV Storage 142 in Fig. 2 to find a plan that has a link or URL to the workflow definition file to be optimized by the Client Interface. For example, Figs. 73A-C depict an excerpt of an exemplary plan definition file 7300 that has a link 7302 in Fig. 73B to “AssemblyWorkflow” definition file 6800 shown in Fig. 68. In this example, to find link 7302, the Client Interface may parse plan definition file 7300 for a task 7340 having a unique identifier set to “NONE” 7304 and having a caption 7306 that is set to the same value as a name 7308 for the plan. Thus, the Client Interface is able to find and recognize that link 7302 matches the location of the workflow that the enterprise affiliate has requested to be optimized and retrieves the plan definition file 7300 for further processing.

In another implementation, the Client Interface may retrieve a plan created from the workflow by using known WebDAV protocol methods, such as “PROPFIND,” to retrieve all plans on WebDAV Storage 142 that have a common WebDAV property set to the link or URL for workflow definition file 6800 in Fig. 68. For example, the Client Interface may retrieve all plans having a WebDAV property named “PROJECT” that is set to link 7302 or “http://localhost:8080/webdav/ProcessGroup2/AssemblyWorkflow.xml.” In this

implementation, when the Client Interface performs the process in Fig. 3 to create each plan from workflow definition file 6800, the Client Interface also creates the WebDAV property named "PROJECT" for each plan definition file and then sets the "PROJECT" property to the URL of workflow definition file 6800. The Client Interface may, however, create the WebDAV property of the plan for storing the URL to the workflow definition file with a name other than "PROJECT."

After retrieving the plan created from the workflow, the Client Interface increments the plan counter (Step 6716 in Fig. 67B). Next, the Client Interface retrieves a task of the plan (Step 6718). In the example shown in Figs. 73A-C, the Client Interface may retrieve a task of the plan by parsing plan definition file 7300 to identify a "Get Parts" task 7310 as shown in Fig. 73A.

After retrieving a task of the plan, the Client Interface determines if a resource was manually assigned to a role of the task (Step 6720). In the example shown in Fig. 73A, to determine if a "MG" resource 7312 (i.e., "Mister Gopher" resource 7110 in Fig. 71) was manually assigned to a "Gopher" role 7314 for "Get Parts" task 7310, the Client Interface retrieves a resource-assignment property 7316 from "Get Parts" task 7310 to identify if the resource-assignment property 7316 is set to "Manual" or "Auto." The Client Interface recognizes that resource-assignment property 7316 is set to "Manual," indicating that "Gopher" role 7314 for "Get Parts" task 7310 has been manually assigned by an enterprise affiliate and another earlier version of plan definition file 7300 is stored on WebDAV Storage 142.

Fig. 74 depicts an excerpt of another exemplary plan definition file 7400 created by the Client Interface from "AssemblyWorkflow" definition file 6800 in Fig. 68, where plan definition file 7400 is an earlier version of plan definition file 7300. When the Client Interface retrieves plan definition file 7400, the Client Interface is able to confirm that plan definition file 7400 is an earlier version of plan definition file 7300 by identifying that "Get Parts" task 7410 has a resource-assignment property 7416 set to "Auto," indicating that "MA" resource 7412 (i.e., "Mister Assembler" resource 7108 in Fig. 71) was automatically assigned to a "Gopher" role 7414 for "Get Parts" task 7410 when plan definition file 7400 was created from "AssemblyWorkflow" definition file 6800 in Fig. 68.

The Client Interface may implement versioning by creating a WebDAV property named "Version" (not shown) for plan definition files 7300 and 7400. The Client Interface may then set the "Version" property to "First" for plan definition file 7400 when file 7400 is created by the Client Interface and set the "Version" property to "Second" for plan definition file 7300

when file 7300 is created by the Client Interface. Thus, in the event that “Gopher” role 7314 of the “Get Parts” task 7310 has been manually reassigned to replacement resource 7312 by an enterprise affiliate using the Client Interface, the Client Interface is able to recognize that plan definition file 7300 is a later version of plan definition file 7400.

5 In another implementation, while performing the process depicted in Fig. 3 above, the Client Interface may record in a manual-assignment log (not shown) for plans created from “AssemblyWorkflow” definition file 7400 the following: a task role (e.g., “Gopher” role 7414 for “Get Parts” task 7410) that has been manually reassigned to another resource, the other resource that replaces an auto-assigned resource, and the auto-assigned resource. In this
10 implementation, the Client Interface need not implement versioning of plan definition files as the Client Interface is able to quickly determine if a resource was manually assigned to a role of the task by referencing the manual-assignment log.

If it was determined that the resource was manually assigned to a role of the task, the Client Interface determines whether the resource replaced an auto-assigned resource (Step
15 6722). In the example shown in Fig. 73 and Fig. 74, the Client Interface may confirm that “MG” resource 7312 replaced an auto-assigned resource 7412 by first retrieving plan definition file 7400, which is an earlier version of plan definition file 7300. The Client Interface may then identify if resource-assignment property 7416 of “Get Parts” task 7410 is set to “Auto,” indicating that “MA” resource 7412 was automatically assigned to “Gopher” role 7414 for “Get
20 Parts” task 7410 when plan definition file 7400 was created from “AssemblyWorkflow” definition file 6800 in Fig. 68. The Client Interface is able to recognize that if resource-assignment property is not set to “Auto” or “Manual,” then the Client Interface did not assign a resource to “Gopher” role 7414.

In another implementation where the Client Interface records each manual assignment as
25 it occurs in the manual-assignment log (not shown) for plans created from “AssemblyWorkflow” definition file 7400, the Client Interface may reference the manual-assignment log to identify if “Gopher” role 7314 for “Get Parts” task 7310 had an auto-assigned resource that was replaced by the manually assigned “MG” resource 7312.

If it is determined that the resource replaced an auto-assigned resource, the Client
30 Interface increments the auto-assignment override counter for the auto-assigned resource (Step 6724). For the example shown in Fig. 74, the Client Interface increments the auto-assignment counter for “MA” resource 7412.

After incrementing the auto-assignment override counter for the auto-assigned resource or after it is determined that the resource did not replace an auto-assigned resource, the Client Interface increments the manual-assignment counter for the resource (Step 6726). For the example shown in Figs. 73A-C, the Client Interface increments the manual-assignment counter for “MG” resource 7312.

After incrementing the manual-assignment counter for the resource, the Client Interface logs or stores reassignment information, such as task role, skill and strength; auto-assigned resource profile; and replacement resource profile (Step 6728). The Client Interface may store the reassignment information so that the Client Interface is able to quickly reference the reassignment information based on the identification of the auto-assigned resource or the manually assigned resource. For example, the reassignment information may be organized in a database table format (not shown) so that each resource associated with each role of the workflow has a corresponding database table. The Client Interface may then store the reassignment information as a record in the database table corresponding to the auto-assigned resource and as a record in the database table corresponding to the manually assigned resource. Thus, the Client Interface is able to determine the percentage of manual-assignments per plans checked for each resource and the percentage of override-assignments per plans checked for each resource.

After storing reassignment information, the Client Interface determines if all tasks of the plan have been checked (Step 6730). The Client Interface may determine that all tasks of the plan have been checked when the task does not have a successor task. For example, the Client Interface is able to recognize that “Get Parts” task 7310 in Fig. 73A has successor 7318 that identifies “Task_20” as the next task in the plan. Thus, the Client Interface recognizes that not all tasks of the plan have been checked. The Client Interface, however, may also determine that all tasks of the plan have been checked when the Client Interface determines that there are no more tasks (i.e., 7310, 7320, 7330, 7340, 7350, 7360 and 7370 in Figs. 73A-C) to be retrieved from plan definition file 7300.

When it is determined that all tasks of the plan have not been checked, the Client Interface retrieves the next task (Step 6732). After retrieving the next task, the Client Interface continues processing at Step 6720.

When it is determined that all tasks of the plan have not been checked, the Client Interface determines if all plans that have been created from the workflow have been checked

(Step 6734). If all plans that have been created from the workflow have not been checked, the Client Interface retrieves the next plan from WebDAV Storage 142 (Step 6736). After retrieving the next plan, the Client Interface continues processing at Step 6716.

When it is determined that all plans that have been created from the workflow have been checked, the Client Interface selects a resource associated with a role of the workflow (Step 6738 in Fig. 67C). The Client Interface may select any resource that has been previously determined by the Client Interface as being assigned to a role of an activity of the workflow. In another implementation, the Client Interface may select a resource that has been assigned to a role of the activity and has been stored as an auto-assigned resource or as manually assigned resource in the reassignment information log.

After selecting the resource associated with a role of the workflow, the Client Interface determines if the maximum percentage of overrides for the resource has been exceeded (Step 6740). To determine if the maximum percentage of overrides for the resource has been exceeded, the Client Interface calculates an actual-percentage-of-overrides for the resource by dividing the number of times that the resource was replaced or overridden by a manually-assigned resource (i.e., value of the auto-assignment counter for the resource) by the number of plans checked (i.e., value of the plan counter) times one hundred (100). The Client Interface then compares the maximum percentage of overrides for the resource with the calculated actual-percentage-of-overrides to determine if the maximum percentage of overrides for the resource has been exceeded. For example, assuming the Client Interface has selected “MA” resource 7412 in Fig. 74, the Client Interface is able to identify that maximum percentage of overrides 7138 of twenty percent (20%) has been exceeded when the Client Interface identifies that “MA” resource 7412 of “Get Parts” task 7410 has been replaced by manually-assigned “MG” resource 7312 in twenty-one (21) out of one hundred plans checked by the Client Interface.

When it is determined that the maximum percentage of overrides for the resource has been exceeded, the Client Interface identifies a skill of the resource that matches a skill of the task role that has been reassigned and identifies a skill of a replacement resource most-often-assigned (Step 6742). Continuing with the previous example, the Client Interface may identify that “MG” resource 7312 in Fig. 73A of “Get Parts” task 7310 is the resource that is most-often-assigned to replace auto-assigned “MA” resource 7412 in Fig.74 of “Get Parts” task 7410. The Client Interface then retrieves role profile 7006 in Fig. 70 for “Gopher” role 7314 in Fig. 73A that has been manually-reassigned to “MG” resource 7312 of “Get Parts” task 7310. The Client

Interface also retrieves resource profile 7102 in Fig. 71 for auto-assigned “MA” resource 7412 in Fig. 74 and resource profile 7104 in Fig. 71 for replacement “MG” resource 7312 in Fig. 73A. The Client Interface is then able to identify that auto-assigned “MA” resource 7412 has skills 7126 in Fig. 71 and 7128 that match “enterprise procurement logistics” skill 7029 in Fig. 70 and “Chicago plant receiving logistics” skill 7030 in “Gopher” role profile 7006. The Client Interface is also able to identify that auto-assigned “MA” resource 7412 has skill 7126 that matches an “enterprise procurement logistics” skill 7148 in Fig. 71 in “MG” resource profile 7104.

Next, the Client Interface notifies an enterprise affiliate of alternative suggestions to improve the profile of the resource (Step 6744). In one implementation, the Client Interface may notify the enterprise affiliate that initiated the request to optimize resource allocation for the “AssemblyWorkflow.” The Client Interface may notify the enterprise affiliate via a dialog box (not shown) displayed by the Client Interface in association with the same user interface that the enterprise affiliate used to initiate the request to optimize the resource allocation.

In another implementation, the Client Interface is able to notify the enterprise affiliate by sending an e-mail message to the enterprise affiliate via network 108 in Fig. 1. The Client Interface is able to identify an e-mail address for the enterprise affiliate by accessing a user profile for the enterprise affiliate that is stored on WebDAV Storage 142 as discussed above. The alternative suggestions to improve the profile of the resource may include removing a skill or decreasing a skill strength in the resource profile that corresponds to the skill of the task role that has been manually reassigned as further discussed below.

After notifying an enterprise affiliate of alternative suggestions to improve the profile of the resource, the Client Interface determines if either of the alternative suggestions is to be implemented (Step 6746). In one implementation, the enterprise affiliate indicates to the Client Interface which, if any, of the alternative suggestions are to be implemented by actuating a button (not shown) next to the alternative suggestion that is displayed by the Client Interface in association with the same user interface that the enterprise affiliate used to initiate the request to optimize the resource allocation. In another implementation, the Client Interface is configured to receive an e-mail reply to the notification sent by the Client Interface and to recognize the alternative suggestion within the reply that the enterprise affiliate has selected to be implemented.

When it is determined that one of the alternative suggestions is to be implemented, the Client Interface determines if the alternative suggestion to implement corresponds to removing the matching skill from the profile of the resource (Step 6748). When it is determined that the alternative suggestion to implement corresponds to removing the matching skill from the profile of the resource, the Client Interface removes the skill from the profile of the resource that matches the skill of the task role (Step 6750). For example, the Client Interface may remove one or both of skills 7126 in Fig. 71 and 7128 in “MA” resource profile that the Client Interface found matched “enterprise procurement logistics” skill 7029 in Fig. 70 and “Chicago plant receiving logistics” skill 7030 in “Gopher” role profile 7006. Thus, when creating a subsequent plan from “AssemblyWorkflow” definition file 6800 and performing resource allocation processing for the plan, the Client Interface recognizes that improved “MA” resource profile does not have skills that match “Gopher” activity role 6832 in Fig. 68. Therefore, the Client Interface does not automatically assign “MA” resource to the “Gopher” role of the new task created from “Get Parts” activity 6804 in Fig. 68.

When it is determined that the alternative suggestion to implement does not correspond to removing the matching skill from the profile of the resource, the Client Interface decreases a skill strength in the profile of the resource to below the strength of the matching skill of the replacement resource (Step 6752). For example, the Client Interface may decrease the strength of skill 7126 in Fig. 71 in “MA” resource profile to below the strength of matching skill 7148 in Fig. 71 of “MG” resource profile 7104. Assuming the Client Interface improves “MG” resource profile 7104 to include all skills in skill set 7018 of “Gopher” role profile as described below, then the Client Interface is able to identify that skill set 7116 in “MG” resource profile 7104 has higher corresponding skill strengths than in skill set 7114 in “MA” resource profile 7102. Thus, in this example, the Client Interface automatically assigns “MG” resource 7110, instead of “MA” resource 7108, to a new task created from “Get Parts” activity 6804 in Fig. 68 by the Client Interface.

When it is determined that the maximum percentage of overrides for resource has not been exceeded, when it is determined that neither of the two alternative suggestions are to be implemented, or after decreasing a skill strength in the profile of the resource to below the strength of the matching skill of the replacement resource, the Client Interface determines if the maximum percentage of manual assignments for the resource has been exceeded (Step 6754 in Fig. 67D). To determine if the maximum percentage of manual assignments for the resource has

been exceeded, the Client Interface calculates an actual-percentage-of-manual-assignments for the resource by dividing the number of times that the resource was manually-assigned to a task role (i.e., value of the manual-assignment counter for the resource) by the number of plans checked (i.e., value of the plan counter) times hundred (100). The Client Interface then compares the maximum percentage of manual assignments for the resource with the calculated actual-percentage-of-manual-assignments to determine if the maximum percentage of manual assignments for the resource has been exceeded. For example, assuming the Client Interface has selected "MG" resource 7312 in Fig. 73A, the Client Interface is able to identify that maximum percentage of manual-assignments 7140 of twenty percent (20%) has been exceeded when the Client Interface identifies that "MG" resource 7312 of "Get Parts" task 7310 has been manually-assigned to replace "MA" resource 7412 in twenty-one (21) out of one hundred plans checked by the Client Interface.

When it is determined that the maximum percentage of manual assignments for the resource has been exceeded, the Client Interface identifies all skills associated with a role for each logged task where the resource was manually assigned (Step 6756). When "MG" resource is selected by the Client Interface, the Client Interface may identify that "Get Parts" task 7310 in Fig. 73A is representative of each task in the reassignment information log where "MG" resource was manually assigned. The Client Interface then identifies that "Gopher" role 7314 in Fig. 73A is assigned to each task in the reassignment information log that corresponds to "Get Parts" task 7310. The Client Interface is then able to identify that "Gopher" role 7314 has skills corresponding to skill set 7018 in Fig. 70 for "Gopher" role profile 7006.

After identifying all skills associated with a role for each logged task where the resource was manually assigned, the Client Interface determines if the resource is missing a skill (Step 6758). To determine if the resource is missing a skill, the Client Interface retrieves the profile for the resource and compares the skill set in the resource profile to the skills or skill identified for the role. Continuing with the above example, the Client Interface retrieves "MG" resource profile 7104. The Client Interface then compares skill set 7116 in resource profile 7104 with skill set 7018 in "Gopher" role profile 7006. The Client Interface is able to recognize that "MG" skill set 7116 is missing "Chicago plant receiving logistics" skill 7030 in "Gopher" skill set 7018.

When it is determined that the resource is missing a skill, the Client Interface generates a suggestion to improve the resource profile by adding the missing skill to the resource profile

(Step 6760). For example, the Client Interface generates a suggestion to improve “MG” profile 7104 by adding the missing “Chicago plant receiving logistics” skill 7030 to “MG” skill set 7116.

When it is determined that the resource is not missing a skill, the Client Interface identifies a highest skill strength for the role from among each auto-assigned resource that was replaced by the resource (Step 6762). For example, the Client Interface may identify that “MA” resource 7412 in Fig. 74 is representative of each auto-assigned resource that was replaced by “MG” resource 7312 in Fig. 73A. The Client Interface is then able to identify that “MA” resource 7412 has skills 7126 and 7128 that match skills 7029 and 7030 in “Gopher” role profile 7006. The Client Interface is able to further identify that “MA” skills 7126 and 7128 have corresponding skill strengths of “6” and “5,” respectively. Thus, the Client Interface identifies the highest skill strengths for “Gopher” skills 7029 and 7030 are “6” and “5,” respectively.

After identifying a highest skill strength for the role from among each auto-assigned resource that was replaced by the resource, the Client Interface generates a suggestion to improve the resource profile by increasing a corresponding skill strength of the resource to above the identified highest skill strength (Step 6764). Continuing with the above example, the Client Interface is able to identify that “MG” skill 7148 corresponds to “MA” skill 7126 and “Gopher” skill 7029 in Fig. 70. The Client Interface then generates a suggestion to improve “MG” resource profile 7104 in Fig. 71 by increasing the strength of “MG” skill 7148 above the identified highest corresponding skill strength of “6.”

After generating a suggestion to add a missing skill to the resource profile or after generating a suggestion to increase a skill strength in the resource profile, the Client Interface notifies the enterprise affiliate of the suggestion to improve the resource profile (Step 6766). The Client Interface may notify the enterprise affiliate via a dialog box (not shown) displayed by the Client Interface in association with the same user interface that the enterprise affiliate used to initiate the request to optimize resource allocation. The Client Interface may also notify the enterprise affiliate by sending an e-mail message to the enterprise affiliate via network 108 in Fig. 1.

After notifying the enterprise affiliate of the suggestion to improve the profile of the resource, the Client Interface determines if the suggested improvement is to be implemented (Step 6768). In one implementation, the enterprise affiliate indicates to the Client Interface if the suggested improvement is to be implemented by actuating a button (not shown) next to the

suggestion that is displayed by the Client Interface in association with the same user interface that the enterprise affiliate used to initiate the request to optimize resource allocation. In another implementation, the Client Interface is configured to receive an e-mail reply to the notification sent by the Client Interface and to recognize a confirmation to implement the suggested improvement within the reply.

When it is determined that the suggested improvement is to be implemented, the Client Interface modifies the profile of the resource to reflect the suggested improvement (Step 6770). For example, when the suggested improvement is to add the missing skill to the resource profile, the Client Interface adds the missing "Chicago plant receiving logistics" skill 7030 to skill set 7116 in "MG" resource profile 7104. When the suggested improvement is to increase the skill strength of the resource profile, the Client Interface increases the strength of "MG" skill 7148 in "MG" profile 7104 above the identified highest corresponding skill strength of "6." Note that the Client Interface identifies that the suggested improvement is to increase the skill strength of the resource profile when skill set 7116 in "MG" resource profile 7104 already has all the skills found in "Gopher" skill set 7018. Thus, when creating a subsequent plan from "AssemblyWorkflow" definition file 6800 and performing resource allocation processing for the plan, the Client Interface recognizes that improved "MG" resource profile 7104 has an improved skill set that matches "Gopher" skill set 7018 and that has higher corresponding skill strengths than "MA" skill set 7114. Therefore, the Client Interface automatically assigns "MG" resource in lieu of "MA" resource to the "Gopher" role of a new task created from "Get Parts" activity 6804 in Fig. 68.

When it is determined that the maximum percentage of manual assignments for the resource has not been exceeded, when it is determined that the suggested improvement is not to be implemented, or after modifying the profile of the resource to reflect the suggested improvement, the Client Interface determines if there are more resources associated with a role of the overflow (Step 6772 in Fig. 67E). If there are more resources associated with a role of the workflow, the Client Interface selects the next resource associated with a role of the workflow (Step 6774). After selecting the next resource, the Client Interface continues processing at step 6740 in Fig. 67C. When it is determined that there are no more resources associated with a role of the workflow, the Client Interface completes processing.

While various embodiments of the application have been described, it will be apparent to those of ordinary skill in the art that many more embodiments and implementations are possible

that are within the scope of this invention. Accordingly, the invention is not to be restricted except in light of the attached claims and their equivalents.